TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

PETRI LAAKKONEN

LASKENTATYÖKALUN KEHITTÄMINEN KANTAVIEN RAKENTEIDEN OPTIMOINTIIN

Diplomityö

Tarkastajat: professori Keijo Ruohonen
yliopistolehtori Sami Pajunen
Tarkastaja ja aihe hyväksytty
Automaatio-, kone- ja materiaalitekniikan tiedekuntaneuvoston kokouksessa 5. kesäkuuta 2013

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Automaatiotekniikan koulutusohjelma
LAAKKONEN, PETRI: Laskentatyökalun kehittäminen kantavien rakenteiden optimointiin
Diplomityö, 132 sivua, 27 liitesivua
Huhtikuu 2014
Pääaine: Teknillinen matematiikka
Tarkastajat: professori Keijo Ruohonen, yliopistonlehtori Sami Pajunen
Avainsanat: Optimointi, simulaatio, metamalli, korvikemalli, vastepintamenetelmä, srsm, lujuuslaskenta

Kova kilpailu on kasvattanut laskennallisesti vaativaa suunnittelua tekevissä yrityksissä tarvetta nopeuttaa suunnitteluprosessia. Täten myös kiinnostus simulaatiopohjaiseen optimointiin ja sen automatisointiin on kasvanut. Tämän työn tavoite on ollut kehittää helppokäyttöinen laskentatyökalu kantavien rakenteiden optimointiin. Suunnitteluprosessia on haluttu automatisoida siten, että parametrisoidulle rakennemallille voidaan hakea paras mahdollinen rakenne työkalun avulla. Laskentatyökaluun implementoitava optimointiproseduuri perustuu Tampereen teknillisen yliopiston tekemään aikaisempaan tutkimukseen lujuuslaskennan automatisoinnista. Siinä optimoidaan vaiheittaisella vastepintamenetelmällä saatua metamallia eli korvikemallia. Esitetyn menetelmän keskeinen idea on, että suunnitteluavaruudesta viritetään vaiheittain aliavaruus aina edellisen optimipisteen ympäriltä ennalta asetetun approksimaatiovirheen sallimissa rajoissa. Joka vaiheessa löydetään tällöin yleensä vähintään lokaali optimi. Työkalun kehittämisessä saadaan tarkempaa tietoa kehitetyn vaiheittaisen vastepintamenetelmän sopivuudesta kantavien rakenteiden optimointiin. Käytännön tehtävien sisältäessä usein diskreettejä suunnittelumuuttujia, selvitetään myös diskreetin optimoinnin ja metamallin yhdistämisen toimivuutta. Työssä laaditaan erityyppisille diskreeteille joukoille matemaattiset mallit diskreettiä optimointia varten. Esitettyä vastepintamenetelmää tutkitaan tässä työssä erilaisilla asetuksilla ja optimointialgoritmeilla.

Työ jakautuu neljään osaan. Ensimmäisessä osassa kerrotaan simulaatiopohjaiseen optimointiin liittyvistä menetelmistä sekä esitellään laskentatyökalun käyttämän vastepintamenetelmän teoriaa ja siihen liittyvää koesuunnittelua. Toisessa osassa käydään läpi optimoinnin kannalta keskeisintä teoriaa ja kerrotaan työssä käytetyistä optimointimenetelmistä. Optimointia käsittelevässä luvussa esitetään myös laaditut mallit diskreeteille tehtäville. Kolmannessa osassa käydään läpi laskentatyökalun toteutusta ja käyttöä. Viimeisessä osassa esitetään ja analysoidaan työkalulla saavutetut tulokset.

Laskentatyökalun käyttäminen vaiheittaisella vastepintamenetelmällä ja optimoinnilla on lukuisten testiprobleemien ja rakennemallien optimoinnin tulosten perusteella tuottanut hyviä tuloksia. Edellytyksenä on, että laskentatyökaluun valitaan sopivat asetukset. Diskreeteille tehtäville laadittujen matemaattisten mallien todettiin toimivan myös käytännössä, joten niitä voidaan hyödyntää yleisesti myös muissa käytännön diskreeteissä optimointiprobleemissa.

Työkalun kehittämistä ja markkinointia yrityksille ja muille toimijoille kannattaa saatujen hyvien tulosten perusteella jatkaa. Jatkokehityksessä työkaluun on hyvä lisätä myös muita simulaatiopohjaisia optimointistrategioita erityyppisiä tehtäviä varten sekä rajapintoja muihin tarvittaviin suunnitteluohjelmiin.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Automation Technology
LAAKKONEN, PETRI: Development of a computational tool for structural optimization
Master of Science Thesis, 132 pages, 27 Appendix pages
April 2014
Major: Technical mathematics
Examiner: Professor Keijo Ruohonen, Associate Professor Sami Pajunen
Keywords: Optimization, simulation, metamodel, surrogate model, response surface methodology, srsm, strength analysis

Hard competition has grown the need of accelerate design process for companies that make computationally demanding design. At the same time the interest in simulation based optimization and its automation has also increased. The goal for this thesis is to develop an easy-to-use computational tool for structural optimization. The objective is to automate the engineering design process so that the tool can be used in the search for the best possible structure of the parametric design model. Implemented optimization procedure of the computational tool is based on the earlier study made by Tampere university of technology on the automation of the strength analysis. Presented method optimizes the metamodel that has been gained with sequential response surface method.
In the design space the metal model is spanned sequentially into an optimally restricted subspace. In these cases one finds at least local optimum at every stage. In the developing of the tool more exact information about the developed method is obtained. When the practical tasks often contain discrete design variables, the use of discrete optimization and metamodel will also be studied. Furthermore, the mathematical models are created in this work to discrete tasks of different types. The presented response surface method is studied with different settings and optimization algorithms in this work.

Thesis has been divided into four parts. The first part is about methods that are related to the simulation based optimization. Furthermore, the theory of the response surface method used by the computational tool and the design of experiments which are related to it are presented. The second part goes through the essential theory involving optimization and the methods that has been used in the work. The models that have been developed for discrete tasks also are presented in this part. Third part tells about the developing and use of the computational tool. Last part presents and analyzes the results gained with the tool.

Based on the results of the optimization of several test problems and design models good results are usually achieved with the computational tool. This requires that computational tool has been used with suitable settings. It was noticed that the mathematical models that have been developed for discrete tasks were operating also in practice. That means that they can be utilized generally also in other practical discrete optimization problems.

Developing and marketing of the computational tool to the companies and other actors is based on these good results highly recommended. In further research it would also be wise to implement other simulation based optimization strategies for diverse tasks and also interfaces to different kinds of design programs.

ALKUSANAT

Tämä diplomityö on tehty Tampereen teknillisen yliopiston teknillisen matematiikan pääaineeseen. Työn teettäjänä toimi Tampereen teknillisen yliopiston Kone- ja tuotantotekniikan laitos.

Haluan kiittää työn erinomaisesta ohjauksesta yliopistonlehtori Sami Pajusta sekä työn tarkastajana toiminutta professori Keijo Ruohosta työhön liittyvistä asiantuntevista kommenteista. Lisäksi haluan kiittää kaikkia muita työhön osallistuneita henkilöitä hyvästä ilmapiiristä ja lukuisista ideoista.

Erityisesti haluan kiittää perhettäni, vaimoa ja lapsia suuresta tuesta, jota ilman en olisi onnistunut yhdistämään opintoja ja töitä sekä perhe-elämää. Haluan kiittää myös kaikki muita läheisiä ja ystäviä, joilta olen saanut tukea ja kannustusta opintojen aikana.

Tampereella 14. huhtikuuta 2014

Petri Laakkonen
Sähköposti: laakkonen.petri@gmail.com

SISÄLLYS

LYHENTEET JA MERKINNÄT

convtol	asetettu konvergenssitoleranssi
FEM-malli	lujuuslaskennassa käytettävä rakennemalli
KKT	Karush-Kuhn-Tucker:n optimaalisuusehdot
LP	lineaarinen optimointitehtävä
Metamalli	korvikemalli
MILP	lineaarinen sekalukutehtävä
MINLP	epälineaarinen sekalukutehtävä
MIQCQP	kvadraattinen sekalukutehtävä kvadraattisilla rajoitusyhtälöillä
MO	monitavoiteoptimointitehtävä
PNS	pienimmän neliösumman menetelmä
PQ	mallin tyyppi: puhtaasti kvadraattinen malli
Q	mallin tyyppi: kvadraattinen malli
QCQP	kvadraattinen tehtävä kvadraattisilla rajoitusyhtälöillä
ROI	tarkasteltava aliavaruus (region of interest)
RSM	vastepintamenetelmä (Response Surface Methodology)
SO	yhden kohdefunktion optimointitehtävä
SQP	kvadraattinen optimointimenetelmä (Sequential quadratic programming)
SRSM	Successive Response Surface Method
TolCon	absoluuttinen virhetoleranssi rajoitusyhtälöille
TolRel	asetettu tavoitevirhe
TSP	kauppamatkustajan ongelma
$\mathbf{0}$	0-vektori
$\mathbf{1}$	1-vektori
α_k	ROI:n säde iteraatiokierroksella k
$\boldsymbol{\beta}$	regressioyhtälön parametrit
$\mathbf{A}$	lineaarisen yhtälörajoitteen matriisi
$\mathbf{b}$	lineaarisen yhtälörajoitteen vektori
B	muuttujan tyyppi: binäärimuuttuja
$\mathbf{B}$	diskreeteistä joukoista $\mathbf{s}_i$ muodostettu matriisi
$\mathbf{c}$	kustannusvektori
C	muuttujan tyyppi: jatkuva muuttuja
$\mathbf{C}$	valintamatriisi diskreeteille arvoille
D	muuttujan tyyppi: riippuva diskreetti muuttuja
$\mathbf{D}$	riippuvuusmatriisi diskreeteille muuttujille
$\mathbf{d}_i$	riippuvuusvektori diskreeteille muuttujille
$\mathbf{E}$	lineaarisen epäyhtälörajoitteen matriisi
$\mathbf{e}$	residuaali
ε	virhetermi

$\overline{\boldsymbol{gvıol}}$	keskiarvo rajoitusyhtälöiden rikkomiselle
$\mathbf{i}$	lineaarisen epäyhtälörajoitteen vektori
I	muuttujan tyyppi: kokonaislukumuuttuja
$\mathbf{l}$	muuttujien alarajat
$\mathbf{I}_n$	identiteettimatriisi
K	riippumattomien joukkojen lukumäärä
L	mallin tyyppi: lineaarinen malli
L	diskreettien arvojen lukumäärä yhteensä vektoreissa $\mathbf{s}_i$
l_i	diskreettien arvojen lukumäärä vektorissa $\mathbf{s}_i$
N	koepisteiden lukumäärä
n	muuttujien lukumäärä
$\mathbf{O}_n$	nollamatriisi
P	riippuvien joukkojen lukumäärä
$\mathbf{r}$	vasteiden suhteelliset virheet
$\mathbf{S}$	$\mathbf{S} \subset \mathbb{R}^n$
S	muuttujan tyyppi: riippumaton diskreetti muuttuja
$\mathbf{s}_i$	diskreetti joukko faktorille x_i
$\mathbf{u}$	muuttujien ylärajat
y	vaste
$\hat{y}$	vasteen approksimaatio
$\mathbf{x}^*$	optimipiste
$\mathbf{X}$	suunnittelumatriisi

1 JOHDANTO

Tampereen teknillisen yliopiston Kone- ja tuotantotekniikan laitoksella on tehty esitutkimusta automaattiseen lujuuslaskentaan liittyen, joista tuloksena on useampi diplomityö ja tieteellinen artikkeli (Heinonen & Pajunen 2011; Pajunen & Heinonen 2013). Viimeisimmässä artikkelissa Pajunen & Heinonen (2013) esitelty menetelmä implementoidaan toteutettavaan laskentatyökaluun. Esitetyssä menetelmässä käytetään vaiheittaista vastepintamenetelmää siten, että approksimoitua mallia varten tarvittava aliavaruus viritetään aina edellisen optimipisteen ympäriltä ennalta asetetun approksimaatiovirheen sallimissa rajoissa. Esitetty menetelmä pohjautuu vastepintamenetelmään Successive Response Surface Method (SRSM) (Roux et al. 1998; Stander & Craig 2002). Laskentatyökalun kehittäminen liittyy laajaan monivuotiseen Tekes-projektiin "Computational methods in mechanical engineering product development (SIMPRO)" (VTT 2012). Työkalun työnimeksi annettiin tämän projektin mukaisesti SimPro.

Työn tavoitteena on kehittää laskentatyökalu kantavien rakenteiden optimointiin ja saada tarkempaa tietoa edellä esitetyn menetelmän sopivuudesta siihen. Päätavoitteena on suunnitella käyttöliittymä, jolla voidaan kutsua ANSYS-ohjelmaa vasteiden laskentaan ja optimoida niiden avulla valittua rakennemallia. Optimointia varten ohjelmaan implementoidaan vastepintamenetelmä, jonka tuottamalla lineaarisella ja kvadraattisella metamallilla eli korvikemallilla voidaan approksimoida vasteita. Saatua metamallia optimoidaan lineaarisilla ja kvadraattisilla optimointialgoritmeilla. Työhön kuuluu myös metamallien ja erilaisten optimointialgoritmien tehokkuuden ja tarkkuuden vertaileminen erilaisilla laskentatyökalun asetuksilla. Työhön otettiin mukaan myös diskreetin optimoinnin ja metamallin yhdistämisen toimivuuden selvittäminen. Esitutkimuksessa Pajunen & Heinonen (2013) on käytetty ainoastaan lineaarista metamallia ja haettu jatkuvaa optimia, jonka ympäriltä on lopuksi haettu kokonaislukuoptimi binäärioptimoinnilla. Esitutkimuksessa laadittua MATLABin laskentakoodia käytetään pohjana laskentatyökalun kehittämisessä muokaten sitä tarpeiden mukaan. Työkalussa tulee olla myös mahdollisuus ajaa tehtäväjonoon lisättyjä optimointitehtäviä. Työkalun tulee myös tuottaa erilaisia raportteja, joiden avulla tuloksia voidaan analysoida. Loppukäyttäjää varten laskentatyökalun löytämä paras ratkaisu on oltava helposti avattavissa käyttöliittymän kautta suoraan ANSYS-ohjelmaan. Lisäksi laskentatyökaluun implementoidaan yliopiston käytössä oleva Techila-laskentaympäristö. Työssä selvitetään myös mitä valmisohjelmia on tarjolla kantavien rakenteiden optimointiin.

Laskentatyökalun toteutuksessa käytetään vain regressioanalyysilla saatua lineaarista ja kvadraattista vastepintaa. Muut metamallit jätettiin tässä vaiheessa työkalusta pois. Työssä kuitenkin esitellään simulaatiopohjaiseen optimointiin liittyviä strategioita, joita

voidaan työkaluun mahdollisesti myöhemmin implementoida. Työkaluun on myöhemmin lisätty myös monitavoiteoptimointi, mutta tässä työssä käsitellään vain yhden kohdefunktion tehtäviä. Työssä ei myöskään tehdä herkkyysanalyysia saaduille malleille. Saatujen tuloksien perusteella tarkastellaan pääasiassa vain valitun mallin, optimointialgoritmin ja laskentatyökalun asetusten vaikutusta kohdefunktion suppenemiseen.

Koska työn päätavoitteena on laskentatyökalun kehittäminen, niin työssä käydään läpi vain lyhyesti koesuunnitteluun, vastepintamenetelmään ja optimointimenetelmiin liittyvä keskeinen teoria. Pääpaino on työssä käytetyissä menetelmissä. Lisäksi matemaattiset todistelut sivuutetaan. Työssä laaditaan myös malli diskreetteihin tehtäviin, joissa muuttujat valitaan äärellisistä diskreeteistä joukoista, siten että eri muuttujat voivat olla riippumattomia tai riippuvia toisistaan.

Työhön on valittu kirjallisuudesta kattava otos erityyppisiä optimointitehtäviä työkalun toimivuuden tarkistamiseksi. Lopuksi työkalulla optimoidaan kahta ANSYS-mallia jatkuvilla muuttujilla ja kokonaislukumuuttujilla.

2 SIMULAATIOPOHJAINEN OPTIMOINTI

Kantavien rakenteiden suunnitteluun liittyvässä optimoinnissa käytetään simulaatiopohjaista optimointia, koska analyyttisia vastefunktioita ei yleensä tiedetä. Tuntemattomien vastefunktioiden ajatellaan olevan niin sanottuja ”black box”-funktioita (Shan & Gary Wang 2010). Tehtävien erilaisesta luonteesta johtuen on kehitetty erilaisia strategioita simulaatiopohjaiseen optimointiin. Strategian valintaan vaikuttaa erityisesti vastefunktioiden ja suunnittelumuuttujien tyyppi, jotka voivat olla jatkuvia tai diskreettejä. Lisäksi jatkuvilla tehtävillä kohdefunktion differentioituvuus vaikuttaa strategian valintaan. Metamalliin perustuvat menetelmät sopivat parhaiten jatkuviin differentioituviin tehtäviin. (Barton & Meckesheimer 2006) Strategioita simulaatiopohjaiseen optimointiin on kehitetty lukuisia, jotka voidaan jakaa seuraaviin menetelmiin (Fu 2002; Baron 2009):

- sijoitus ja valinta (ranking and selection)
- metamalliin perustuvat menetelmät (metamodel based methods)
- gradientti-pohjaiset menetelmät (gradient based procedures)
- satunnaishaku (random search)
- stokastinen vastine (stochastic counterpart / sample path optimization)
- metaheurestiikka
- joukko todennäköisyysjakaumia käyttäviä mallipohjaisia menetelmiä.

Tässä työssä käytetään metamalleihin kuuluvaa vaiheittaista regressioanalyysiin perustuvaa vastepintamenetelmää, josta kerrotaan luvussa 2.1. Tärkeimpiä metamalliin perustuvia menetelmiä ovat (Barton & Meckesheimer 2006):

- (vaiheittainen) regressioanalyysiin perustuva vastepintamenetelmä (response surface methodology)
- neuroverkot (neural network)
- spatiaalinen korrelaatio eli Kriging-menetelmä (spatial correlation)
- radiaalikantafunktio (radial basis function)
- regressiosplini (regression spline).

Metamallien avulla voidaan approksimoida tuntemattomia ”black box”-funktioita. Edellisistä vastepintamenetelmä ja Kriging-menetelmä ovat käytetyimpiä kantavien rakenteiden suunnittelussa (Pajunen & Heinonen 2013). Suunnittelussa käytettävistä metamalleista tehdyn kyselyn perusteella annetaan suosituksena, että deterministiset probleemat joissa on yli 50 faktoria, kannattaisi mallintaa neuroverkoilla. Raportissa suositellaan Kriging-mallin käyttämistä probleemoihin joissa faktoreita on alle 50. Vastepintamenetelmää ei suositella käytettävän jos mallinnettavat vasteet ovat hyvin epälineaarisia tai jos muuttujia on paljon. Vastepintamenetelmää suositellaan käytettävän vain jos probleemassa faktoreita on alle 10 ja ne ovat melko hyvin käyttäytyviä. (Simpson et al.

2001; Ramu et al. 2010) ANSYS-ohjelmassa puolestaan suositellaan korkeintaan 20 faktoria vastepintamenetelmän kanssa. Metamalliin perustuva optimointi voidaan jakaa seuraaviin päävaiheisiin, joista kerrotaan seuraavissa luvuissa tarkemmin (Ramu et al. 2010):

- luodaan tarvittava data koesuunnittelulla
- sovitetaan metamalli havaintoihin
- optimoidaan metamallia.

Näitä vaiheita voidaan myös toistaa tarkkuuden saamiseksi, yleensä siten että jatketaan parhaimman ratkaisun lähiympäristöstä. Metamalleilla optimointi voidaan jakaa vielä globaalin ja lokaalin mallin sovituksen strategiaan. Globaalissa sovituksessa sovitetaan malli koko suunnitteluavaruudelle, jota sitten optimoidaan. Lokaalissa sovituksessa malli sovitetaan jokaisella iteraatiokierroksella edellisen optimipisteen lähiympäristöön siten, että tehtävää rajoittaa kuitenkin asetetut globaalit rajat. Tätä vaiheittain tapahtuvaa strategiaa käytetään erityisesti vastepintamenetelmän kanssa, koska vastepintamenetelmän tarkkuus ei yleensä riitä approksimoimaan vastefunktioita kuin pienestä suunnitteluavaruuden osasta. (Barton 2009)

Luvussa 3.5 esitellään muutamaa simulaatiopohjaista optimointia käyttävää valmisohjelmaa, joita voidaan käyttää myös kantavien rakenteiden optimointiin. Kirjallisuudessa Fu (2002) sekä Barton & Meckesheimer (2006) hämmästelevät sitä, ettei kaupallisiin valmisohjelmistoihin olla vaiheittaista vastepintamenetelmää implementoitu, vaikka vaiheittaista vastepintamenetelmää on tutkittu paljon ja se kuuluu yleisimpiin kirjallisuudessa esiintyviin simulaatiopohjaisiin optimointimenetelmiin. Työssä kehitettävällä laskentatyökalulla on täten uutuusarvoa jos vastaavaa menetelmää hyödyntävää kaupallista ohjelmaa ei ole edelleenkään saatavilla.

2.1 Vastepintamenetelmä

Tässä työssä käytetään vastepintamenetelmällä saatua metamallia, jonka tarkkuus riippuu tarkasteltavan suunnitteluavaruuden koosta, suunnittelumatriisin optimaalisuudesta ja käytetyn polynomin asteesta. Suunnittelumatriisin optimaalisuuteen vaikuttaa koepisteiden lukumäärä ja miten ne on valittu. (Ramu et al. 2010) Koesuunnittelusta kerrotaan tarkemmin seuraavassa luvussa.

Vastepintamenetelmässä käytetään yleensä 1. ja 2. asteen polynomeja vastefunktioiden approksimointiin. Korkeammalla asteluvulla saataisiin enemmän tarkkuutta, mutta tällöin myös tarvittavien koepisteiden määrä kasvaa huomattavasti. Lisäksi korkea asteluku voi aiheuttaa voimakasta värähtelyä. (Ramu et al. 2010; Huang et al. 2011) Kehitettävässä laskentatyökalussa käytetään edellä mainituista syistä vain 1. ja 2. asteen polynomimallia vasteiden approksimointiin. Vastepintamenetelmässä polynomimallien parametrit $\boldsymbol{\beta}$ lasketaan yleensä pienimmän neliösumman regressiomenetelmällä (PNS) (Ramu et al. 2010).

Ensimmäisen asteen eli lineaarinen regressiomalli voidaan esittää

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \varepsilon, \tag{1}$$

joka on vektorimuodossa

$$y = \mathbf{x}\boldsymbol{\beta} + \varepsilon, \tag{2}$$

missä

$$\begin{aligned} \mathbf{x} &= [1 \quad x_1 \quad \cdots \quad x_k] \text{ ja} \\ \boldsymbol{\beta} &= [\beta_0 \quad \beta_1 \quad \cdots \quad \beta x_k]^T, \end{aligned} \tag{3}$$

jossa **x** on suunnitteluvektori ja **β** on mallin parametrit. Lineaarinen regressiomalli voi sisältää myös vuorovaikutustermit, tällöin (Myers et al. 2009)

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \sum_{i<j}^{k} \sum_{=2} \beta_{ij} x_i x_j + \varepsilon. \tag{4}$$

Lineaarista regressiomallia, jossa on ristikkäistermit mukana ei tässä työssä kuitenkaan käytetä. Tarkempi malli saadaan käyttämällä toisen asteen eli kvadraattista regressiomallia joka voidaan kirjoittaa (Myers et al. 2009)

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \sum_{i<j}^{k} \sum_{=2} \beta_{ij} x_i x_j + \sum_{j=1}^{k} \beta_{jj} x_j^2 + \varepsilon. \tag{5}$$

Kvadraattista mallia ilman ristitermejä sanotaan puhtaasti kvadraattiseksi. Vastepintamenetelmässä käytetään usein toisen asteen regressiomallia, koska sillä saadaan esitettyä hyvin erilaisia funktioita ja sen parametrit on helppo estimoida pienimmän neliösumman sovituksella. Yhtälöissä esiintyvää satunnaisvirhettä ε ei esiinny deterministisissä tapauksissa. (Myers et al. 2009) Regressioyhtälö esitetään yleensä yleisessä matriisimuodossa

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{6}$$

missä

$$\begin{aligned} \mathbf{y} &= [y_1 \quad y_2 \quad \cdots \quad y_N]^T, \\ \boldsymbol{\varepsilon} &= [\varepsilon_1 \quad \varepsilon_2 \quad \cdots \quad \varepsilon_N]^T, \\ \boldsymbol{\beta} &= [\beta_0 \quad \beta_1 \quad \cdots \quad \beta x_k]^T \text{ ja} \end{aligned} \tag{7}$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{ik} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Nk} \end{bmatrix}.$$

Suunnittelumatriisi **X**, jossa on koepisteitä N kappaletta, tuotetaan jollain koesuunnittelumenetelmällä, esimerkiksi seuraavassa luvussa kuvatulla D-optimaalisella suunnittelulla. Koepisteiden lisäksi suunnittelumatriisissa on **1**-vektori mallin vakiotermejä varten. (Myers et al. 2009)

Pienimmän neliösumman sovituksessa haetaan **β**-vektorin estimaatille **b** parametrit siten, että (Walpole et al. 2007; Myers et al. 2009)

$$\mathbf{L} = (\mathbf{y} - \mathbf{Xb})^T(\mathbf{y} - \mathbf{Xb}) = \mathbf{y}^T\mathbf{y} - 2\mathbf{b}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{Xb} \tag{8}$$

minimoituu. Ottamalla tästä gradientti **b**:n suhteen ja merkitsemällä oikea puoli **0**-vektoriksi saadaan

$$-2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{Xb} = \mathbf{0}, \tag{9}$$

joka sievenee normaaliyhtälöksi

$$\mathbf{X}^T\mathbf{Xb} = \mathbf{X}^T\mathbf{y}, \tag{10}$$

josta voidaan ratkaista haettu **b** jos matriisi $\mathbf{X}^T\mathbf{X}$ on kääntyvä ja $N \geq k + 1$. Ehtojen täyttyessä **β**-vektorin estimaatti

$$\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{11}$$

ja sovitettu regressiomalli

$$\hat{\mathbf{y}} = \mathbf{Xb}. \tag{12}$$

MATLAB-ohjelmassa on useampi tapa ratkaistua mallin parametrit. Kehitettävässä laskentatyökalussa käytetään MATLABin suosittelemaa ”\”-operaattoria (mldivide), joka valitsee automaattisesti tilanteeseen parhaiten sopivimman algoritmin. Mallin sovitus tehdään helposti komennolla b=X\Y, jossa X on datamatriisi ja Y havaintomatriisi tai havaintovektori. Y:n ollessa matriisi, tehtävässä on useampi vastefunktio, jolloin tuloksena on matriisi parametreista. MATLABilla pystytään siis ratkaisemaan useamman vastefunktion parametrit yhdellä komennolla. (Matlab 2013)

Käytännön tehtävissä on usein useampia vastefunktioita, joista osa voi toimia optimointitehtävän rajoitefunktioina. Useamman vastefunktion mallintamiseksi kerrallaan ratkaistaan parametrimatriisi

$$\mathbf{B} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{13}$$

ja sovitettu regressiomalli

$$\hat{\mathbf{Y}} = \mathbf{XB}. \tag{14}$$

Saadulla korvikemallilla voidaan siis ennustaa vasteita tai sitä voidaan optimoida. Optimoiva malli saadaan poimimalla lasketusta **B**-matriisista kohde- ja rajoitefunktiot, jotka ovat matriisin pystyvektoreina eli sarakkeina. Parametrivektoreiden ensimmäisessä alkiossa on mallin vakiotermi, tämän jälkeen tulevat lineaariset termit, sitten vuorovaikutusten termit ja lopuksi neliölliset termit. Ylimääräiset valittuun malliin kuulumattomat termit jäävät pois. Järjestys on sama kuin suunnittelumatriisin riveillä. Koesuunnittelu-luvussa on kerrottu tarkemmin miten termit tulevat suunnittelumatriisiin. Optimointifunktiota varten rajoitefunktion vakiotermi siirretään yleensä rajoitusyhtälön toiselle puolelle, koska useimmat optimointifunktiot sen niin vaativat. Myös kohdefunktio annetaan usein optimointifunktiolle ilman vakiotermiä. Kohdefunktion vakiotermi lisätään vain lopuksi saatuun optimiarvoon. Optimointitehtävään lisätään tarvittaessa vielä esimerkiksi muuttujien ja rajoitusyhtälöiden rajat. Optimoinnista on kerrottu tarkemmin luvussa 3.

Mallin tarkkuus riippuu luvun alussa mainituista asioista, kuten käytetyn polynomin asteesta. Malli toimii yleensä parhaiten suunnitteluavaruuden aliavaruuden sisällä johon se on viritetty. Todellisten havaintojen ja mallin antaman arvojen ero eli residuaali

$$\mathbf{e} = \mathbf{y} - \hat{\boldsymbol{y}}. \tag{15}$$

Korvikemallin hyvyyttä arvioidaan tässä työssä residuaalin lisäksi suhteellisen virheen kautta. Laskentatyökalu tallentaa raporttiin iteraatiokierroksilla saadut tarkat ja mallin antamat arvot ja laskee edellä mainitut. Suhteellista virhettä käytetään aina uuden iteraatiokierroksen aliavaruuden rajojen määrittämiseen luvussa 4.1.2 kuvatulla tavalla. Tällä suhteellisen virheen tarkastelulla pyritään pitämään implementoidussa vaiheittaisessa vastepintamenetelmässä vasteet käypänä ja tavoitteena on, että kohdefunktio suppenee ilman isompaa värähtelyä.

Vastepintamenetelmää on tutkittu paljon ja siitä on tehty useita muunnoksia. Laskentatyökaluun implementoitava menetelmä on lähinnä Success Response Surface Method -menetelmää (SRSM). Yhteistä menetelmillä on se, että käytetään D-optimaalista suunnittelua vastepintamenetelmän kanssa ja aliavaruuden kokoa säädetään adaptiivisesti jollain kriteerillä iteraatiokierrosten aikana, lähtien liikkeelle käyttäjän antamasta aloituskoosta. (Stander & Craig 2002) Implementoitavassa menetelmässä kriteerinä on

edellä mainittu suhteellinen virhe, joka lasketaan jokaiselle vasteelle ja suurinta saatua virhettä käytetään seuraavan iteraatiokierroksen aliavaruuden koon määrittämiseen.

2.2 Koesuunnittelu

Koepisteet voidaan tuottaa usealla eri koesuunnittelumenetelmällä. Vastepintamenetelmässä käytetään yleisesti D-optimaalista koesuunnittelua sekä CCD- ja Box-Behnken-koesuunnittelumenetelmää (Ramu et al. 2010). Vastepintamenetelmän tarvitseman suunnittelumatriisin muodostamiseen käytetään esitutkimuksessa käytettyä D-optimaalista koesuunnittelumenetelmää siten, että koepisteitä lasketaan käytetyn polynomin asteen tarvitsema minimimäärä. Minimimäärä koepisteitä tarkoittaa myös sitä, ettei mallissa ole sovituksesta aiheutuvaa virhettä. (Pajunen & Heinonen 2013) Minimimäärä koepisteitä saadaan käytetyn polynomiapproksimaation kertoimien lukumäärästä, johon on lisätty vielä vakiotermi. Kaavoissa 16-17 (Redhe et al. 2002) kerrotaan miten voidaan laskea minimimäärä koepisteitä 1. ja 2. asteen polynomimalleille. Kaavoissa n on faktoreiden lukumäärä.

Lineaariselle eli 1. asteen polynomimallille ilman ristitermejä koepisteiden lukumäärä

$$N = n + 1, \tag{16}$$

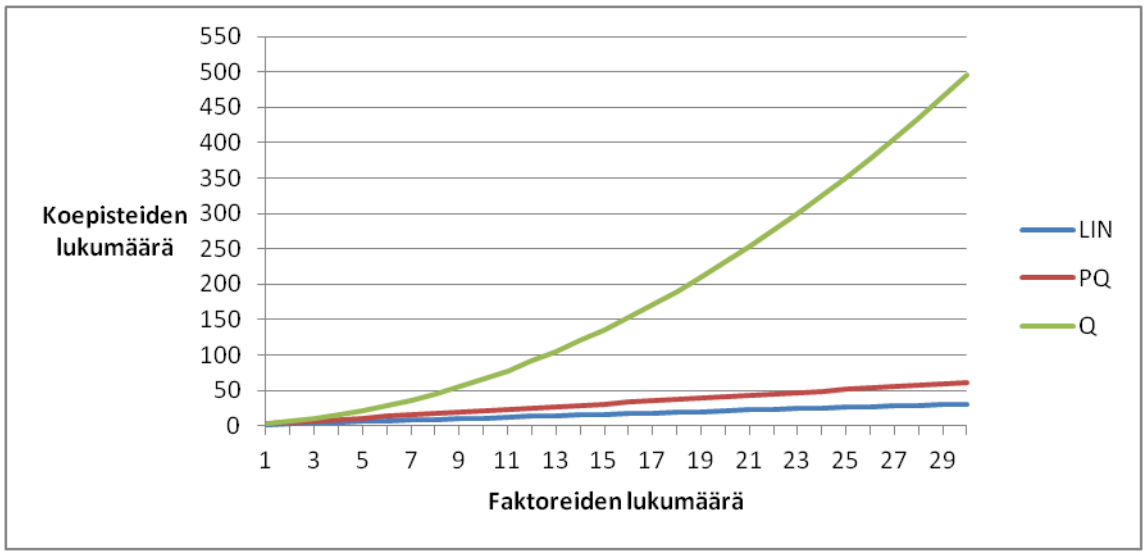
kvadraattiselle eli 2. asteen polynomimallille koepisteiden lukumäärä

$$N = (n + 1)(n + 2)/2 \tag{17}$$

ja jos kvadraattisesta mallista poistetaan ristitermit niin koepisteiden lukumäärä

$$N = 2n + 1. \tag{18}$$

Kuvasta 1 nähdään miten faktoreiden lukumäärä vaikuttaa laskettavien koepisteiden määrään. Kvadraattisissa tehtävissä laskettavien koepisteiden lukumäärä kasvaa huomattavasti faktoreiden lukumäärän kasvaessa. Esimerkiksi Tulokset-luvussa esiteltävän lastiluukun optimoinnissa on 17 suunnittelumuuttujaa, jolloin kvadraattisen malliin tarvitaan jopa 171 koepistettä. Täten, muuttujien määrän ollessa suuri, kannattaa harkita lineaarisen mallin käyttämistä, varsinkin jos ei ole käytössä hajautettua laskentaympäristöä. Kuvassa 1 on merkitty lineaarista LIN, kvadraattista Q ja puhtaasti kvadraattista tehtävää PQ.



Kuva 1 *Koepisteiden lukumäärä faktoreiden lukumäärän funktiona.*

Suunnittelun sanotaan olevaan optimaalinen kun se on jollain kriteerillä mitaten paras mahdollinen. D-optimaalisuus on näistä kriteereistä käytetyimpiä. (Montgomery 2003) D-optimaalinen suunnittelu eroaa muista koesuunnittelumenetelmissä siinä, ettei se vaadi ortogonaalista suunnittelumatriisia. Tähän kuitenkin usein päästään sillä D-optimaalinen suunnittelu saadaan minimoimalla parametrien kovarianssia, johon päästään maksimoimalla iteratiivisesti determinanttia $|\mathbf{X}^T\mathbf{X}|$. (Matlab 2013) Lineaarialgebrasta muistetaan myös, että matriisi on lineaarisesti riippuva jos ja vain jos sen determinantti on nolla. Koska malli sovitetaan havaintoihin pienimmän neliösumman regressiomenetelmällä (PNS), täytyy matriisin $\mathbf{X}^T\mathbf{X}$ olla ei-singulaarinen eli kääntyvä, josta kertoo myös nollasta poikkeava determinantti.

D-optimaalisen suunnittelun luomiseen on MATLAB-ohjelmassa kaksi funktiota rowexch ja cordexch, joita molempia käytetään tässä työssä. Funktiot palauttavat koematriisin ja suunnittelumatriisin ja varoittavat jos matriisi $\mathbf{X}^T\mathbf{X}$ on singulaarinen tai lähellä sitä. Suunnittelumatriisien riveille tulee kvadraattista mallia varten termit seuraavassa järjestyksessä (Matlab 2013):

- vakio termi
- lineaariset termit 1,2,…,*n*
- vuorovaikutustermit järjestyksessä (1, 2), (1, 3), ..., (1, *n*), (2, 3), ..., (*n*-1, *n*)
- neliölliset termit järjestyksessä 1,2,…,*n*.

Muista malleista jäävät pois niihin kuulumattomat termit. Koematriisista puuttuu suunnittelumatriisissa mukana oleva vakiotermi. Koematriisia voidaan käyttää suoraan vasteiden laskemiseen ja suunnittelumatriisia tarvitaan mallin sovitukseen. Kumpaankin funktioon annetaan argumenttina faktoreiden ja koepisteiden lukumäärä, suunnitteluväli, regressiomallin tyyppi sekä yritteiden lukumäärä. Oletuksena molemmat funktiot hakevat lineaariselle mallille kahden ja kvadraattiselle mallille kolmen tason suunnittelun. Lineaarista mallia varten suunnittelumatriisiin tulee aina koepistekombinaatioita faktoreiden suunnitteluvälin ala- ja ylärajoista. Kvadraattisten mallien kanssa mukaan tulee keskimmäinen piste rajojen väliltä. Kvadraattisiin malleihin voidaan asettaa lisää

tasoja tarpeiden mukaan. Tasojen lisäämisellä voidaan saada sovitettua tarkempi malli. Tässä työssä on käytetty tasoihin oletusasetusta, koska aliavaruuden koko pysyy käytetyssä vaiheittaisessa vastepintamenetelmässä usein melko pienenä. Useamman tason käyttö kvadraattisten mallien kanssa voisi olla kannattavaa kun tarkasteltava aliavaruus on hieman suurempi. Tätä kannattaisi tutkia ja kokeilla työkalun jatkokehityksessä. Yritteiden lukumäärä tarkoittaa sitä, että kuinka monta kertaa haetaan optimaalista suunnittelua. MATLABin rowexch- ja cordexch-funktiot hakevat ratkaisua iteratiivisesti satunnaisella alkusuunnittelulla, jolloin saatu suunnittelumatriisi eroaa yleensä joka ajolla. Kumpikaan funktioista ei välttämättä löydä aina parasta mahdollista suunnittelua. Saatu ratkaisu voi olla vain lokaalisti optimi. Yritteiden määrää kasvattamalla voidaan löytää parempi ratkaisu, sillä MATLAB vertaa yritteillä saatuja determinantin arvoja keskenään ja palauttaa parhaan löydetyn suunnittelumatriisin. Funktioissa olennaisena erona on se, että rowexch-funktio hakee parasta suunnittelua tekemällä suunnittelumatriisikandidaatteihin rivinvaihto-operaatioita, kun taas cordexch vaihtaa suunnittelumatriisista yksittäisiä elementtejä. Pienemmän hakualueen takia cordexch-funktio jää helpommin lokaaliin minimiin, joten rowexch-funktiolla voidaan päästä parempaan tulokseen. D-optimaalinen suunnittelu voidaan tehdä myös candexch-funktiolla jolle annetaan itse määritelty kandidaattimatriisi. candexch-funktio käyttää rivinvaihto-operaatioita vastaavasti kuin rowexch. Kandidaattimatriisin käytöllä voidaan siis itse määritellä kaikki koepistevaihtoehdot, joista sitten muodostetaan D-optimaalinen suunnittelu. Kandidaattimatriisia voidaan käyttää esimerkiksi riippuvien diskreettien joukkojen kanssa, joista kerrotaan luvussa 3.4. MATLABin dokumentaatiossa todetaan, että rowexchin muistin tarve on huomattavasti suurempi kuin cordexch-funktiolla jos kandidaattimatriisi on suuri. (Matlab 2013) Tämä on huomattu myös käytännössä laskentatyökalun kehittämisen aikana. Suunnittelumuuttujien lukumäärän kasvaessa muistin kulutus kasvaa rowexch-funktiolla huomattavasti, kuten myös ratkaisuun kulunut aika. Työkalun kehittämiseen käytetystä tietokoneesta, jossa oli 16 Gt työmuistia, loppui tehtyjen testien mukaan muisti lineaarisella mallilla jo 25 suunnittelumuuttujalla. Tästä syystä laskentatyökalussa käytetään molempia funktioita siten, että lineaarisen mallin kanssa käytetään cordexch-funktiota kun suunnittelumuuttujia on yli 20 ja kvadraattisten mallien kanssa jo kun muuttujia on yli 10. Muuten käytetään rowexch-funktiota. Suunnittelumatriisien laskennan hitaudesta johtuen yritteiden määrä on laskentatyökalussa rajattu kahteen. Laskentatyökalussa funktioiden yritteiden määrä voisi olla suhteessa suunnittelumuuttujien lukumäärään. Yritteiden määrä voisi olla myös käyttäjän säädettävissä käyttöliittymän kautta. Lisäksi funktio, jolla koematriisi luodaan, voisi olla käyttäjän valittavissa, koska tietokoneen työmuistin määrä vaihtelee eri loppukäyttäjillä. Funktioiden toiminnallisuudesta kerrotaan tarkemmin MATLABin dokumentaatiossa (Matlab 2013).

Suunnittelumatriisin luomisen jälkeen tehdään kokeet eli lasketaan vasteiden arvot koepisteissä. Tuloksena on havaintovektori **y** kun vastefunktioita on vain yksi ja havaintomatriisi **Y** jos vastefunktioita on useampi. Kokeiden jälkeen voidaan sovittaa metamalli tehtyihin havaintoihin.

2.3 Hajautettu laskenta

Hajautetussa laskennassa ideana on se, että laskennallisesti vaativa tehtävä voidaan jakaa useampaan osaan ja laskea yhtä aikaa useammalla tietokoneella. Tampereen teknillisen yliopiston käytössä on Techila-järjestelmä, joka perustuu siihen, että käytetään asiakkaan vapaiden tietotokoneiden resursseja hyödyksi. Tietokoneet ovat verkon kautta yhteydessä Techilan palvelimeen, joka jakaa työt vapaille koneille. Loppukäyttäjä saa tehtävien valmistuttua tulokset omalle koneelle. (Techila 2014)

Techila tukee useita ohjelmointirajapintoja, kuten tässä työssä käytettyä MATLAB-ohjelmaa. Laskentatyökaluun lisätään käyttäjälle mahdollisuus valita se, että ajetaanko tehtävät paikallisesti vai käytetäänkö Techila-järjestelmää. Käyttäjällä tulee olla käytössään vain Techila-oikeudet ja määriteltynä tietokoneet, joissa on tarvittavat ohjelmat. Järjestelmän ylläpitäjä antaa nämä oikeudet ja määrittelee laskentaympäristön. Kantavien rakenteiden optimoinnissa tarvitaan ainakin ANSYS- ja SolidWorks-ohjelmaa vasteiden laskentaan. (Techila 2014)

Hajautetun laskennan avulla voidaan säästää aikaa huomattavasti. Simulaatiopohjaisessa optimoinnissa lasketaan paljon vasteita esimerkiksi metamallin sovittamista varten. Yhden vasteen laskeminen voi kestää useita minuutteja ja jos koepisteitä on paljon, niin laskenta voi kestää huomattavan pitkään yhdellä tietokoneella. Edellisen luvun kuvassa 1 havainnollistettiin kuinka koepisteiden lukumäärä kasvaa suhteessa suunnittelumuuttujien lukumäärään. Hajauttamalla laskennan Techilan avulla laskettavaksi, tehtävään kulunut aika vähenee suoraan suhteessa vapaiden tietokoneiden määrään. Laskentatyökalun kehittämisen aikana koepisteet lähtivät keskimäärin 20 koneelle laskettavaksi, jolloin tehtävän sai parhaimmillaan ratkaistua muutamassa minuutissa.

3 OPTIMOINTI

Simulaatiopohjaiseen optimointiin liittyy useita laskennallisia haasteita. Käytännön optimointiongelmat kuten kantavien rakenteiden optimointiongelmat ovat lähes aina epälineaarisia ja niissä on usein diskreettejä suunnittelumuuttujia, jolloin ne ovat luonteeltaan MINLP-luokkaan kuuluvia tehtäviä eli epälineaarisia sekalukutehtäviä. Lisäksi vastefunktiot eivät yleensä ole konvekseja, joka vaikeuttaa optimointitehtävää entisestään. (Simpson et al. 2001; Burer & Letchford 2012) Optimointitehtävä voi olla myös stokastinen, jolloin ainakin osa suunnittelumuuttujista on probabilistisia (Rao 2009). Tässä työssä keskitytään vain deterministisiin tehtäviin. Lisäksi tavoitteet voivat olla keskenään ristiriitaisia, jolloin puhutaan monitavoiteoptimoinnista, jota tässä työssä ei käsitellä. Tässä työssä käsitellään tehtäviä, joissa vain yhtä vastetta minimoidaan, muiden vasteiden toimiessa tehtävän rajoitteina. Analyysin laskeminen FEM-rakennemallille vasteiden saamiseksi voi olla myös erittäin hidasta jossain tapauksissa. Edellä mainituista syistä rakenteiden optimoinnissa ei välttämättä löydetä globaalia optimia, mutta usein löydetään silti sitä lähellä oleva lokaali optimi.

Metamallien avulla epälineaarisia vastefunktioita voidaan approksimoida, esimerkiksi vastepintamenetelmällä käyttämällä 1. ja 2. asteen polynomiapproksimaatioita, jolloin tehtävä voidaan muotoilla joko lineaariseksi tai kvadraattiseksi tehtäväksi, joihin löytyy tehokkaita ratkaisualgoritmeja. Epälineaarisia optimointialgoritmeja voidaan käyttää myös lineaaristen ja kvadraattisten tehtävien ratkaisemiseen, mutta ne eivät ole niin tehokkaita kuin lineaarisiin ja kvadraattisiin tehtäviin tarkoitetut algoritmit (Antoniou & Lu 2007). Kvadraattisissa tehtävissä tehokkuuteen ja tarkkuuteen vaikuttaa olennaisesti se, että onko tehtävä konveksi. Konveksisuus vaikuttaa kvadraattisissa tehtävissä algoritmin valintaan, sillä osa niistä pystyy ratkaisemaan vain konvekseja tehtäviä. (Lorenz 2010; Konnik 2013; Matlab 2013)

Algoritmien valintaan yleisesti liittyen ovat Wolpert & Macready (1997) esittäneet niin sanotun ”ei ilmaista lounasta” -teoreemaan, jonka mukaan kaikki algoritmit ovat yhtä hyviä, kun ajatellaan kaiken tyyppisiä ongelmia. Tämä tarkoittaa sitä, että toiset algoritmit ovat parempia tietyissä ongelmissa ja kaikille algoritmeille voi löytyä ne ongelmat joille juuri se sattuu olemaan tehokkain. (Wolpert & Macready 1997)

Algoritmin valinnassa tulee täten huomioida tehtävän tyyppi ja edellä mainitut asiat. Laskentatyökalua varten algoritmin tulee pystyä ratkaisemaan valittu metamalli, joka voi olla lineaarinen tai kvadraattinen, joten algoritmille on voitava antaa myös kvadraattiset rajoitefunktiot. Kvadraattinen tehtävä on tällöin muotoa QCQP. Algoritmien valinnassa otettiin huomiin myös se, että se on helppo implementoida MATLABin laskentakoodiin.

Seuraavaan taulukkoon 1 on koottu vaatimukset täyttävät algoritmit, jotka työkaluun implementoidaan. Taulukossa on kerrottu mitä optimointimenetelmää ne käyttävät ja minkälaisiin tehtäviin niitä käytetään tässä työssä. Niiden käyttämästä päämenetelmästä kerrotaan alaluvuissa. Algoritmit ovat usein muunnoksia jostain tunnetusta optimointimenetelmästä. Optimointifunktioissa voi olla automaattinen algoritmin valinta tehtävän tyypin mukaan tai niissä voi olla ratkaisuna se, että algoritmeja ajetaan peräkkäinen joista valitaan paras ratkaisu. Lisäksi käytetään hybridiratkaisuja, joissa käytetään useampaa menetelmää. (Gurobi 2014; Matlab 2013) Laskentatyökalussa käytetään Gurobin kanssa automaattista asetusta algoritmin valintaan. Luvussa 4.2 kerrotaan tarkemmin implementoiduista algoritmeista ja niiden valintakriteereistä.

Taulukko 1 *Laskentatyökaluun valitut algoritmit ja niiden käyttämät optimointimenetelmät (Matlab 2013; OPTI 2014; Gurobi 2014).*

Algoritmi	Optimointimenetelmä	Metamallin tyyppi Lineaarinen / kvadraattinen
MATLAB linprog - interior point - simplex - active set	Sisäpiste Simplex Aktiivijoukko	Lineaarinen
Gurobi - primal simplex - dual simplex - barrier	Primaali Simplex Duaali Simplex Sisäpiste	Lineaarinen tai kvadraattinen
Scip - branch and bound	Hajota ja rajoita	Kvadraattinen

Seuraavaksi kerrotaan optimointitehtävän muodostamisesta ja siihen liittyvistä keskeisistä asioista. Tämän jälkeen esitellään käytettyjä optimointimenetelmiä.

3.1 Optimointitehtävä

Optimointitehtävä voidaan esittää seuraavassa yleisessä muodossa

$$\begin{gathered} \min\ f(\mathbf{x}) \\ g_i(\mathbf{x}) \leq 0, i = 1, \dots, m \\ h_i(\mathbf{x}) = 0, i = 1, \dots, l \\ \mathbf{x} \in \mathbf{S} \subset \mathbb{R}^n, \end{gathered} \tag{19}$$

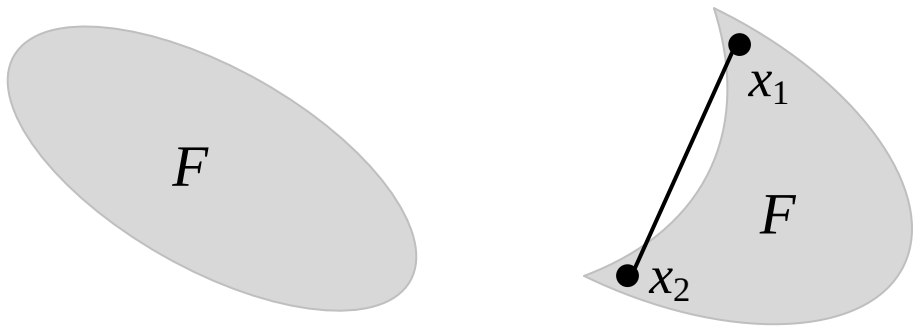
jossa minimoitava kohdefunktio on f, epäyhtälörajoitteiden funktioita ovat g_i ja yhtälörajoitteiden funktioita ovat h_i. Monitavoitetehtävässä kohdefunktioita on useampi. Tavallisesti joukko **S** sisältää muuttujien ala- ja ylärajat. Optimointitehtävän tavoite on siis löytää arvot muuttujille **x**, siten että kohdefunktion arvo minimoituu rajoitteiden ollessa

voimassa. Optimointitehtävän käyväksi ratkaisuksi sanotaan pistettä $\mathbf{x} \in \mathbf{S}$, jossa kaikki rajoitusyhtälöt ovat voimassa. Käypä joukko koostuu kaikista käyvistä pisteistä. Tehtävän parasta löydettyä ratkaisua sanotaan optimiratkaisuksi, jota merkitään tässä $\mathbf{x}^*$ ja sen arvoa $f(\mathbf{x}^*)$. (Bazaraa et al. 2006)

Optimointitehtävänä voi olla myös kohdefunktion maksimointi eli max $f(\mathbf{x})$. Minimointitehtävä voidaan aina helposti muuttaa maksimoinniksi ja toisin päin, sillä max $f(\mathbf{x})$ on sama kuin min $-f(\mathbf{x})$. (Lorenz 2010)

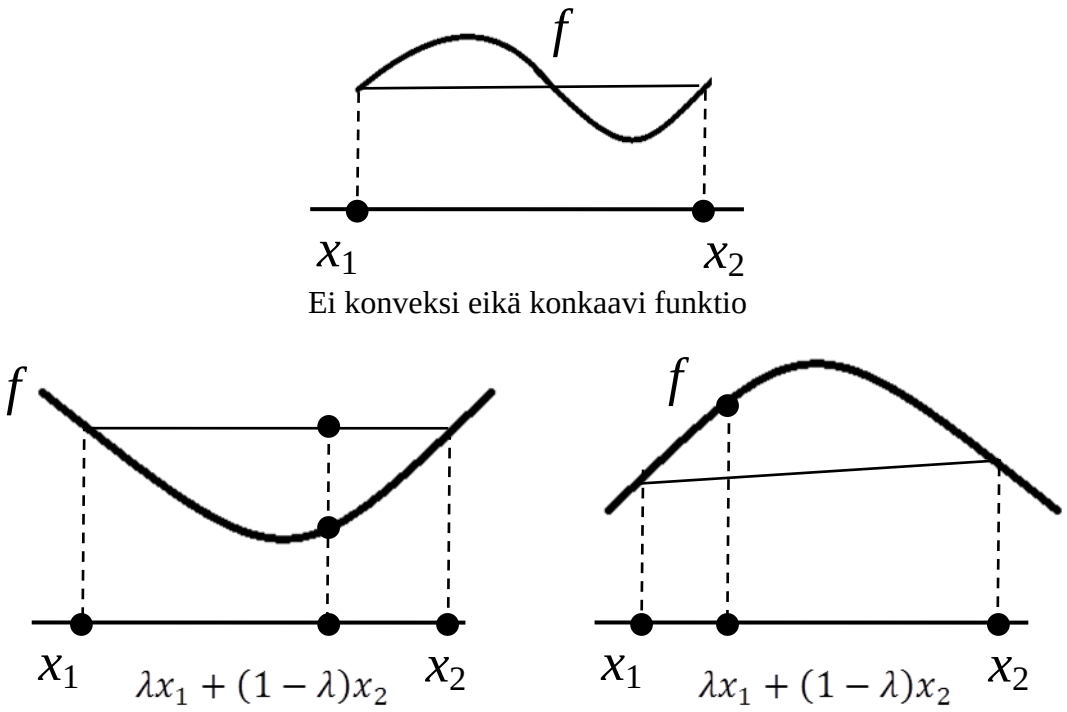
Tärkeä käsite optimoinnissa on tämän työn kannalta konveksisuus, koska se on useille kvadraattisille algoritmeille vaatimuksena. Keskeinen teoria käsitellään tässä kuitenkin vain hyvin lyhyesti. Kirjassa Bazaraa et al. (2006) käsitellään teoriaa hyvin kattavasti.

Tehtävä on konveksi kun sillä on konveksi kohdefunktio $f(\mathbf{x})$ ja konveksi käypä alue. Tehtävän ollessa konveksi, seuraa sitä se, että kaikki lokaalit minimit ovat myös globaaleja. Lineaarisissa tehtävissä kohdefunktio ja käypä joukko ovat aina konvekseja. Epälineaarisissa tehtävissä näin ei välttämättä ole. Joukko $F \in \mathbb{R}^n$ on konveksi jos ja vain jos joukon mitkä tahansa kaksi pistettä voidaan yhdistää suoralla viivalla, joka myös kuuluu joukkoon. Tämä voidaan ilmaista myös, että jos $x_1, x_2 \in F$, niin $\lambda x_1 + (1-\lambda)x_2 \in F$ kaikilla $\lambda \in [0,1]$. (Bazaraa et al. 2006; Lorenz 2010) Kuvassa 2 havainnollistetaan tätä. Oikean puoleinen käypä joukko ei ole konveksi, koska välijana ei kuulu käypään joukkoon.



Kuva 2 *Konveksi ja epäkonveksi joukko.*

Funktio puolestaan on konveksi ei tyhjässä joukossa F jos $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$ toteutuu kaikille $x_1, x_2 \in F$ ja jokaiselle $\lambda \in [0,1]$. Funktiota sanotaan konkaaviksi, jos $-f$ on konveksi joukossa F. Kuvassa 3 havainnollistetaan funktion konveksisuutta. (Bazaraa et al. 2006)



Kuva 3 *Konveksi ja konkaavi funktio (Bazaraa et al. 2006).*

Toinen tärkeä tieto optimointitehtävissä on se, että ovatko funktiot siinä kahdesti differentioituvia. Tämä on myös vaatimuksena useille algoritmeille. Kahdesti differentioituvalle funktiolle voidaan laskea Hessen matriisi, jota käytetään optimoinnissa hyödyksi laskennan lisäksi konveksisuuden selvittämiseen ja optimaalisuuden tarkastamiseen. (Bazaraa et al. 2006)

Funktio on konveksi jos Hessen matriisi on positiivisesti semidefiniitti käyvässä joukossa. Symmetrisesti, jos funktio on konkaavi, niin Hessen matriisi on negatiivisesti semidefiniitti. Matriisin definiittisyys on helppo laskea ominaisarvojen avulla. Matriisi on positiivisesti semidefiniitti kun sen kaikki ominaisarvot ovat positiivisia. Matriisi on positiivisesti definiitti kun sen kaikki ominaisarvot ovat aidosti positiivisia. (Bazaraa et al. 2006)

Hessen matriisin ominaisarvojen avulla voidaan päätellä myös epälineaarisessa rajoittamattomassa optimoinnissa, että onko löydetty optimi minimi vai maksimi, sijoittamalla optimipiste Hessen matriisin. Hessen matriisin ominaisarvojen ollessa positiivisesti definiitti on kyseessä lokaali minimi ja negatiivisesti definiitissä tapauksessa kyseessä on lokaali maksimi. Ominaisarvojen ollessa positiivisia ja negatiivisia, niin kyseessä on satulapiste. Epälineaarisissa rajoitetuissa tehtävissä optimaalisuusehtona käytetään usein Karush-Kuhn-Tucker:n ehtoja eli KKT-ehtoja. KKT-ehto on riittävä ehto konveksille probleemalle. (Bazaraa et al. 2006)

Vaiheittaisessa vastepintamenetelmässä approksimoidusta kvadraattisesta mallista voi tulla epäkonveksi jollain iteraatiokierroksella, vaikka tehtävä olisi alun perin konveksi käyvässä joukossa, johtuen siitä että malli sovitetaan joka kierroksella uudestaan.

Tämä huomattiin kun laskentatyökalulla käytettiin Gurobia kvadraattisen metamallin kanssa. Gurobi ei osaa ratkaista epäkonveksia kvadraattista tehtävää, jolloin optimointi pysähtyy virheilmoitukseen. Scip-algoritmi puolestaan selviää myös epäkonveksista tehtävästä, johtuen erityyppisestä optimointimenetelmästä.

Duaalisuus on myös tärkeä käsite optimoinnista, joten sitä käsitellään hieman lineaarisen optimoinnin luvussa. Duaalisuutta käytetään hyödyksi useiden menetelmien kanssa, koska sillä saadaan yleensä parempia tuloksia aikaan. Duaali-pohjaisessa menetelmässä alkuperäistä tehtävää sanotaan primaaliksi ja sen muunnosta duaaliksi. Esimerkiksi paljon käytetyille Simplex- ja sisäpistemenetelmille on kehitetty duaaliversiot. (Griva et al. 2009)

3.2 Lineaarinen optimointi

Lineaarisessa optimointitehtävässä kohdefunktio ja rajoitefunktiot ovat lineaarisia. Lineaarinen optimointitehtävä (LP) esitetään usein muodossa

$$\begin{aligned} \min\ & f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \\ & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{Ex} = \mathbf{i} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \end{aligned} \tag{20}$$

jossa **l** ja **u** ovat suunnittelumuuttujien ala- ja ylärajat.

Lineaarisessa optimointitehtävässä käypä joukko on konveksi monitahokas, jonka yli lineaarista kohdefunktiota minimoidaan. Lineaarisen optimoinnin teoria tunnetaan hyvin ja siihen on kehitettyä useita menetelmiä. Lineaarisen optimoinnin tehokkuuden takia monet probleemat muotoillaan lineaariseksi tehtäväksi tai approksimoidaan lineaariseksi. Lisäksi lineaarista optimointia käytetään hyödyksi vaikeiden tehtävien ratkaisemisessa. (Bazaraa et al. 2006; Antoniou & Lu 2007)

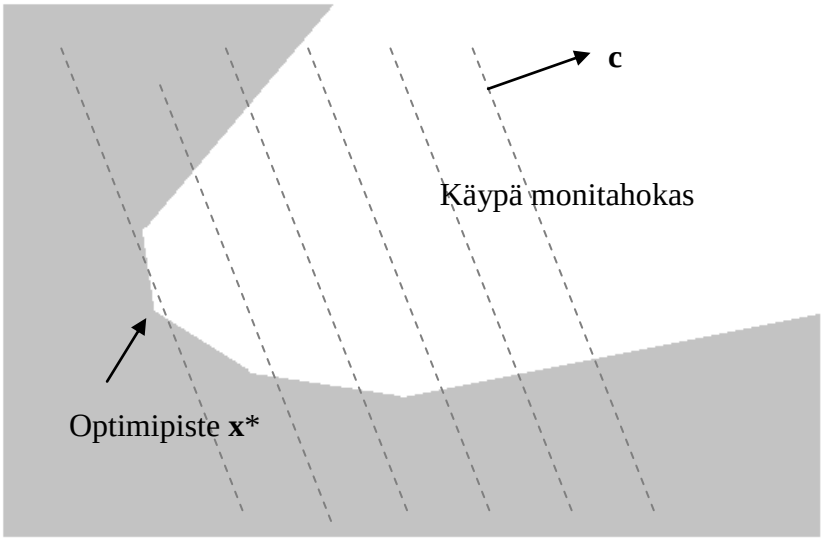
Duaalisuutta hyödyntämällä voidaan kehittää tehokkaampia algoritmeja. Duaalitehtävän keskeinen informaatio on se, että sillä saadaan ala- ja ylärajat kohdefunktiolle. Edistyneet algoritmit hyödyntävät primaali- ja duaali-informaatiota. Epälineaarisilla tehtävillä on saman optimiarvon antama duaali olemassa vain tiettyjen konveksisuusolettamuksien ja rajoitusyhtälöiden laatuvaatimusten täyttyessä. Niiden täyttyessä on mahdollista ratkaista primaalitehtävä epäsuorasti duaalitehtävän kautta. Edellä mainittujen vaatimusten tarkastelu sivuutetaan tässä. Kaikilla lineaarisilla optimointitehtävillä puolestaan on duaali. (Bazaraa et al. 2006; Griva et al. 2009) Duaalisuus on helppo esittää lineaaristen tehtävien kanssa, joten olkoon lineaarisen tehtävän primaali (Griva et al. 2009)

$$\begin{aligned} \min\ & z = \mathbf{c}^T\mathbf{x} \\ & \mathbf{Ax} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{21}$$

jolloin sen duaali on

$$\begin{aligned} \max\ & w = \mathbf{b}^T\mathbf{y} \\ & \mathbf{A}^T\mathbf{y} \leq \mathbf{c} \\ & \mathbf{y} \geq \mathbf{0}. \end{aligned} \tag{22}$$

Seuraavaksi esitellään lyhyesti yleisiä kirjallisuudessa esiintyviä ja valmisohjelmissa, kuten MATLABissa käytettäviä lineaarisiin optimointiin käytettäviä optimointimenetelmiä. Simplex-menetelmä on tunnetuimpia menetelmiä lineaariseen optimointiin. Menetelmä kehitettiin jo vuonna 1947, mutta se on silti pysynyt käytetyimpänä menetelmistä sen tehokkuuden takia. (Antoniou & Lu 2007) Simplex kuuluu yleiseen rajoitettujen optimointitehtävien luokkaan aktiivijoukkomenetelmät. Aktiivijoukko (Active-set) on myös oma menetelmänsä, joka muistuttaa Simplex-menetelmää. Simplex-menetelmän perusidea on se, että liikutaan käyvän alueen reunalla ja haetaan monitahokkaasta käypä kulmapiste, jonka jälkeen tarkastetaan pisteen optimaalisuus. Jos piste täyttää optimaalisuusehdot, niin lopetetaan, muuten haetaan suunta, johon kohdefunktion arvo pienenee ja jatketaan suuntaan kunnes saavutaan käyvän alueen reunalle. Tämän jälkeen aloitetaan alusta etsimällä käypä kulmapiste. Simplex-menetelmän tuottama optimiratkaisu on siis käyvän joukon jokin kulmapiste. Ratkaisuna voi olla myös reuna, jolloin saadaan joukko optimipisteitä. Korkeammissa dimensioissa ratkaisujoukko voi olla myös esimerkiksi jokin pinta. (Nocedal & Wright 1999) Kuvasta 4 nähdään lineaarisen tehtävän monitahokkaan muodostama käypä alue, jonka kulmapisteistä Simplex-algoritmi hakee minimiä. Kuvasta nähdään myös lineaarisen kohdefunktion **c** kasvusuunta ja optimipiste **x***.



Kuva 4 *Lineaarinen optimointitehtävä (Nocedal & Wright 1999).*

Aktiivijoukkomenetelmiä käytetään rajoitteisissa tehtävissä yleisesti, etenkin lineaarisissa ja kvadraattisissa tehtävissä. Se on pohjana esimerkiksi kvadraattisille tehtäville

tarkoitetulle SQP-menetelmälle. (Nocedal & Wright 1999) Aktiivijoukkomenetelmän perusideana on se, että jokaisella iteraatiolla valitaan joukko rajoitteita, jonka oletetaan olevan aktiivinen optimipisteessä. Tämän jälkeen minimoidaan kohdefunktiota tämän joukon yli siten, että rajoitteet ovat yhtälörajoitteita. Kaikki muut rajoitteet jätetään tehtävästä väliaikaisesti pois, mutta rikkoa niitä ei kuitenkaan saa. Tämän jälkeen tarkastetaan optimaalisuus ja mikäli ei ole optimi, niin tiputetaan tietyillä kriteereillä mahdollisesti rajoitteita pois muodostetusta joukosta ja liikutaan suuntaan, johon kohdefunktio pienenee, kunnes vastaan tulee rajoitteet, jonka jälkeen aloitetaan alusta. Piste, jonka ympärille rajoitteet aina iteraatioissa muodostetaan, ei tarvitse olla kulmapiste. (Griva et al. 2009)

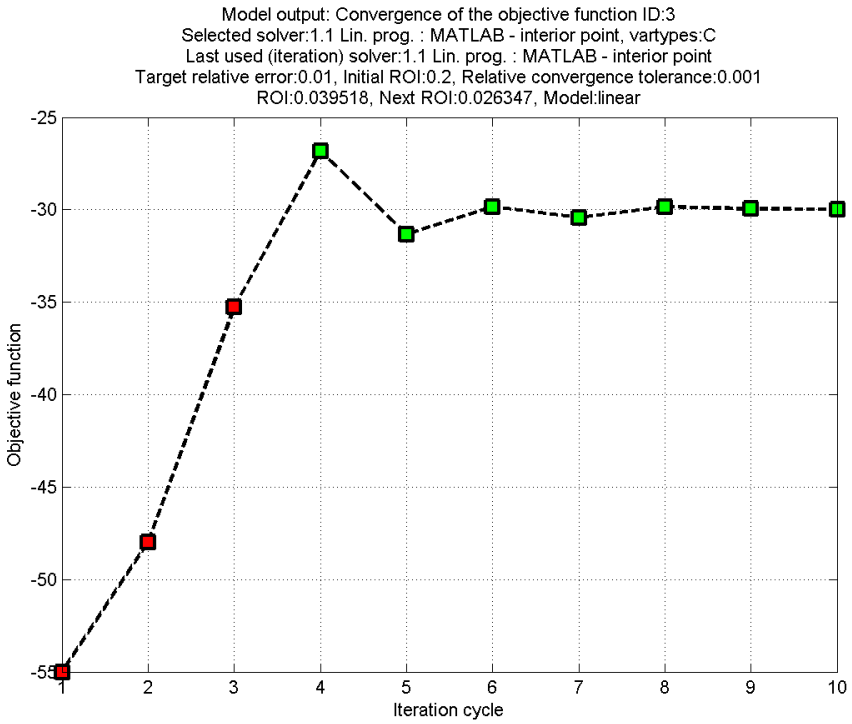
Sisäpistemenetelmät on kolmas yleinen kokoelma menetelmiä. Ne pohjautuvat Karmarkarin alkuperäiseen algoritmiin vuodelta 1984. Ne soveltuvat erityisen hyvin erittäin suuriin tehtäviin, joissa sen on todettu olevan huomattavasti nopeampi kuin Simplex. (Rao 2009) Nimi tulee siitä, että sisäpistemenetelmässä käypien pisteiden hakusuunta lähtee suoraan käyvän alueen sisältä. Tämä tekee menetelmän iteraatiosta kalliimman eli raskaamman laskea, mutta sillä saadaan merkittävä parannus ratkaisun joka iteraatiolla. Simplex-menetelmä puolestaan tarvitsee yleensä paljon laskettavia iteraatioita, mutta menetelmässä iteraatio saadaan laskettua nopeasti. Lisäksi sisäpistemenetelmät eroavat edellisistä siinä, että siinä voidaan käyttää epälineaarisen optimoinnin menetelmiä myös lineaarisiin tehtäviin, kuten Newtonin menetelmää optimaalisuuden tarkastamiseen ja itse optimointiin jotain epälineaariseen optimointiin tarkoitettua algoritmia. (Griva et al. 2009)

Merkittävä asia sisäpistemenetelmässä on se, että sillä voidaan löytää ratkaisu myös lähtien käyvän alueen ulkopuolelta. Tästä on huomattavaa etua, koska usein käytännön tehtävissä on vaikeaa löytää käypää aloituspistettä. (Wright 1997) Tämän työn esitutkimuksesta laaditussa artikkelissa Pajunen & Heinonen (2013) kerrotaan, että vastepinnan optimoinnissa on käytetty Simplex-menetelmää ja saatu ratkaisu myös epäkäyvällä aloituspisteellä. Tämä ei kuitenkaan ole mahdollista, sillä Simplex vaatii käyvän alkupisteen. Kuten edellä kerrottiin, niin Simplex etsii käyvän joukon sisältä ensin käyvän alueen reunan, josta haetaan sitten käypää nurkkapistettä (Bazaraa et al. 2006). Esitutkimuksessa optimointiin käytettiin MATLABin linprog-funktiota, jonka oletettiin ilmeisesti käyttävän oletuksena Simplex-algoritmia. Näin ei kuitenkaan ole, sillä linprog-funktion oletuksena on sisäpistemenetelmään perustuva Interior point -algoritmi, joka selviää siis myös epäkäyvästä aloituspisteestä. Kuitenkaan käypää ratkaisua ei voida löytää, jos käypä joukko on tyhjä. MATLABin Interior point -algoritmi palauttaa siinä tapauksessa parhaiten käyvän pisteen, kun algoritmin lopetusehdot täyttyvät. (Matlab 2013) Implementoidussa vaiheittaisessa vastapintamenetelmässä voi käydä juuri näin, kun lähdetään epäkäyvästä pisteestä. Vaihettaisessa vastepintamenetelmässä iterointikierrokset kuitenkin etenevät, koska MATLABin Interior point -algoritmi palauttaa parhaiten käyvän ratkaisun. Algoritmin palauttama ratkaisu ei välttämättä ole kuitenkaan parempi kuin edellinen piste. Tästä voi seurata vasteen voimakasta värähtelyä iterointien aikana, josta on esimerkkejä Tulokset-luvun testiprobleemissa. MATLABin Simp-

lex-algoritmi puolestaan palauttaa virheilmoituksen, jos käypä joukko on tyhjä. Lisäksi MATLABin Simplex-algoritmi palauttaa tyhjän muuttujavektorin. Tästä seuraa vaiheittaisessa vastepintamenetelmässä iteroinnin keskeytys, koska pistettä ei löytynyt jatkoa varten. Tämän asian ollessa laskentatyökalun kannalta tärkeä, niin tarkastellaan siitä esimerkkiä.

Esimerkki 1

Katsotaan yksinkertainen esimerkki laskentatyökalulla saadusta ratkaisusta, jossa on lähdetty epäkäyvästä alkupisteestä. Kuvasta 5 nähdään metamallin approksimoiman kohdefunktion suppeneminen. Testiprobleemaa TP12 (Hock & Schittkowski 1981) on tässä optimoitu käyttämällä lineaarista metamallia ja MATLABin Interior point -algoritmia. Punaiset pisteet kuvaajassa ovat epäkäypiä pisteitä. Tehtävän ”black box”-funktiot näytetään luvussa 5.2.4. Tässä tarkastellaan vaiheittaisella vastepintamenetelmällä saatua ensimmäistä optimointitehtävää, jonka tulos on kuvan 5 kuvaajan toinen piste.

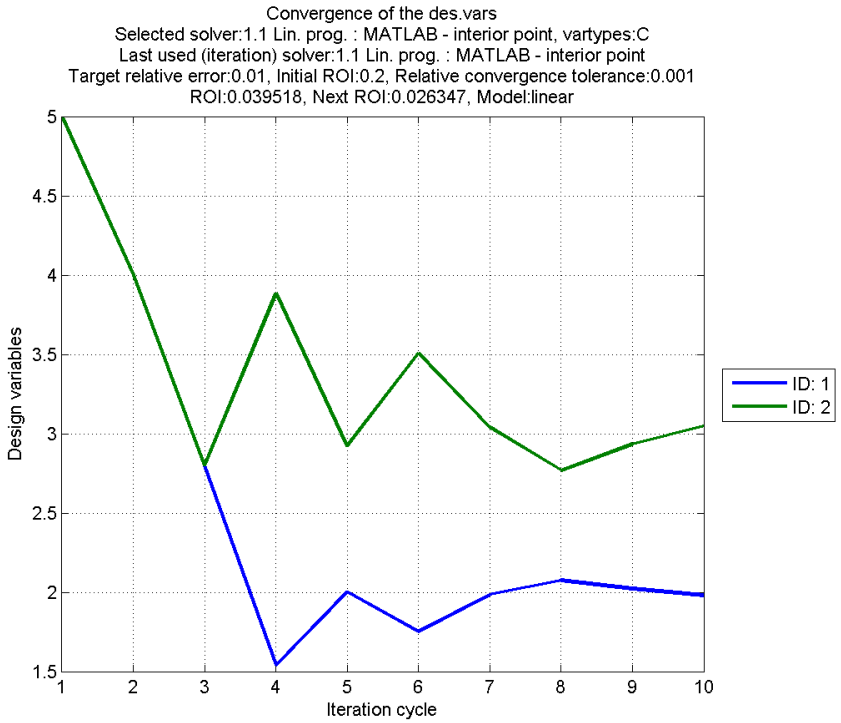


Kuva 5 *Kohdefunktion suppeneminen lineaarisella metamallilla ja Interior point -algoritmilla, kun lähdetään epäkäyvästä alkupisteestä.*

Tehtävään on annettu alkupisteeksi (5, 5) ja ottamalla alkupiste aliavaruuden keskipisteeksi, on saatu aliavaruus, jonka rajat molemmille muuttujille on [4, 6]. Optimoitavassa metamallissa on vain yksi epäyhtälörajoite. Lineaariseksi optimointitehtäväksi saadaan

$$\begin{aligned} \min\ & f(\mathbf{x}) = -6x_1 - x_2 - 20 \\ & 40x_1 + 10x_2 \leq 145 \\ & 4 \leq \mathrm{x}_1, x_2 \leq 6, \end{aligned} \tag{23}$$

josta nähdään selvästi, että käypä joukko on tyhjä ja se, että parhaiten käypä piste muuttujien rajojen sisällä on (4, 4). Tämän pisteen palauttaa myös MATLABin Interior Point -algoritmi, kuten kuvasta 6 nähdään. Kuvista nähdään, että käyvälle alueelle on päästy vasta neljännellä iteraatiokierroksella. Tässä tehtävässä on käynyt siinä mielessä hyvin, että suppeneminen on edennyt oikeaan suuntaan. Kokemukseen perustuen, näin ei välttämättä aina käy, kun lähdetään epäkäyvästä pisteestä.



Kuva 6 *Muuttujien suppeneminen lineaarisella metamallilla ja Interior point -algoritmilla, kun lähdetään epäkäyvästä alkupisteestä.*

MATLABin Active set -algoritmin selviämistä epäkäyvästä alkupisteestä ei tässä työssä selvitetty, koska Interior point -algorimilla saatiin hyviä tuloksia aikaan. Aktiivijoukkomenetelmä ei vaadi käyvän alueen reunaa, kuten Simplex, joten menetelmän puolesta se voisi olla mahdollista.

3.3 Epälineaarinen optimointi

Epälineaarisen tehtävän muoto on esitetty edellä kaavassa (19). Kuten pääluvun alussa todettiin, niin epälineaariset tehtävät ovat usein erittäin vaikeita ratkaista. Epälineaarisesta optimoinnista erikoistapaus on kvadraattinen muoto, jonka yleisessä muodossa (QP) on kvadraattinen kohdefunktio ja lineaariset rajoitteet. Tällöin voidaan käyttää sopivilla muunnoksilla lineaarisia optimointimenetelmiä tehtävän ratkaisemiseksi. Tämä perusmuotoinen kvadraattinen tehtävä on epälineaarista tehtävistä helpoimpia ratkaista, etenkin jos sen kohdefunktio on myös konveksi (Rao 2009; Griva et al. 2009). Yleinen muoto kvadraattiselle tehtävälle on

$$\begin{aligned} \min\ & f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} \\ & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbf{S} \subset \mathbb{R}^n, \end{aligned} \tag{24}$$

jossa **Q** on symmetrinen kvadraattiset termit sisältävä matriisi. **Q**-matriisin ominaisarvoista nähdään suoraan onko funktio konveksi. Sen ollessa positiivisesti semidefiiniitti, niin funktio on konveksi ja täten tehtävä on huomattavasti helpompi ratkaista. (Bazaraa et al. 2006) Kvadraattiset termit sisältävä matriisi täytyy muodostaa algoritmeja varten. Neliölliset termit tulevat **Q**-matriisin diagonaalille, jonka jälkeen matriisi täydennetään ristitermeillä. (Matlab 2013; OPTI 2014; Gurobi 2014)

Kun tehtävään lisätään myös kvadraattiset rajoitusyhtälöt, niin tehtävä vaikeutuu huomattavasti. Kvadraattisia tehtäviä kvadraattisilla rajoitusyhtälöillä (QCQP) pidetään yhtenä kaikkein vaikeimmista tehtävistä ratkaista. (Linderoth 2005) Tämä voidaan esittää seuraavassa muodossa

$$\begin{aligned} \min\ & f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} \\ & g_i(\mathbf{x}) \leq 0, i = 1, \dots, m \\ & \mathbf{x} \in \mathbf{S} \subset \mathbb{R}^n, \end{aligned} \tag{25}$$

jossa rajoitefunktiot g_i ovat kvadraattisia. Tehtävässä voi olla myös lineaarisia rajoitusyhtälöitä.

Epälineaarisiin tehtäviin on kehitetty useita algoritmeja, joista monet perustuvat lineaarisen optimoinnin yhteydessä mainittuihin sisäpiste- ja aktiivijoukkomenetelmiin (Nocedal & Wright 1999; Matlab 2013). Erityisesti kvadraattisiin optimointitehtäviin on kehitetty useita tehokkaita algoritmeja, mutta osa niistä sopii vain yleisen (24) kvadraattisen tehtävän optimointiin, kuten MATLABin quadprog-funktio. Kyseinen funktio käyttää samoihin optimointimenetelmiin perustuvia menetelmiä kuin mitä käyttää MATLABin linprog-funktio. (Matlab 2013)

QP- ja QCQP-tehtävissä voi olla vaatimuksena se, että tehtävän tulee olla konveksi. Näin on esimerkiksi Gurobissa. Siinä voidaan käyttää kuitenkin kvadraattisia rajoitteita. Gurobissa käytetään QCQP-tehtäviin sisäpistemenetelmää, mutta jos tehtävässä on kokonaislukumuuttujia, niin käytetään lisäksi duaali-simplex -menetelmää. (Gurobi 2014)

MATLABin epälineaarisiin tehtäviin tarkoitettuun fmincon-funktiosta mainittakoon, että siihen voidaan valita SQP-menetelmä, jota pidetään yhtenä tehokkaimpana menetelmällä ratkaista epälineaarisia rajoitteita sisältäviä tehtäviä. SQP-menetelmä (Sequential quadratic programming) eli toistettu kvadraattinen optimointi on iteratiivinen menetelmä joka ratkaisee kvadraattisia alitehtäviä. SQP-menetelmä perustuu aktiivijoukkomenetelmään. (Nocedal & Wright 1999)

Perinteisten optimointimenetelmien lisäksi viime vuosina on tullut joukko uudentyyppisiä heurestisia optimointimenetelmiä, jotka sopivat käytännön monimutkaisiin

tehtäviin, joissa perinteiset menetelmät ovat tehottomia. Näihin kuuluvat muun muassa geneettinen optimointi, simuloitu jäähdytys ja neuroverkkoihin perustuvat menetelmät. (Rao 2009) Seuraavassa luvussa mainitaan vielä Branch and bound eli hajota ja rajoita -menetelmä, jota laskentatyökalun Scip-algoritmi käyttää epälineaarisen sekalukutehtävän optimointiin. Se osaa ratkaista myös epäkonveksin tehtävän. (OPTI 2014)

3.4 Diskreetti optimointi

Diskreetillä optimoinnilla tarkoitetaan tässä optimointitehtäviä, joissa muuttujat voivat saada kokonaislukuarvoja tai diskreettejä reaalilukuarvoja äärellisestä joukosta. Tehtävät jotka sisältävät pelkästään kokonaislukumuuttujia, sanotaan kokonaislukutehtäviksi. Jos taas tehtävässä on jatkuvia ja kokonaislukumuuttujia niin kyseessä on sekalukutehtävä. Kokonaislukuja sisältävät tehtävät ovat yleensä erittäin vaikeita ratkaista. (Wolsey 1998)

Etenkin lineaarisiin sekalukutehtäviin on kehitetty useita erilaisia malleja, joiden laskennallisesta vaativuudestakin on tietoa. Niitä voidaan käyttää hyödyksi omaa optimointimallia laadittaessa. Tunnetuimpia mallinnettuja probleemia on kauppamatkustajan ongelma (TSP). Siinä on tehtävänä kiertää tietyt kaupungit siten, että jokaisessa kaupungissa saa käydä vain kerran. Kyseessä on kombinatorinen tehtävä, joka on erittäin vaikea ratkaista kun kohteita on paljon. Kauppamatkustajan ongelma on hyvin tyypillinen käytännön ongelma. Samanlaiseen optimointiongelmaan päästään esimerkiksi kun halutaan minimoida komponenttien ladontaan kulunut aika. Muita tunnettuja malleja ovat esimerkiksi selkäreppuongelma (Knapsack problem) ja kohdistustehtävä (assignment problem). (Wolsey 1998)

Kokonaislukuoptimointiin kehitetyissä menetelmissä hyödynnetään usein jatkuville tehtäville tarkoitettuja menetelmiä. Menetelmien välivaiheissa voidaan esimerkiksi tehdä lineaarinen malli, jota optimoidaan jatkuvana jollain lineaarisella algoritmilla. (Nocedal & Wright 1999) Sekaluku- ja kokonaislukutehtävien ratkaisemisessa käytetään usein Gomoryn leikkaustasomenetelmää (cutting plane) ja hajota ja rajoita -menetelmää (branch and bound) (Rao 2009). Laskentatyökalun Scip-algoritmi käyttää hajota ja rajoita -menetelmää, joten kerrotaan siitä hieman.

Hajota ja rajoita -menetelmää pidetään hyvin tehokkaana menetelmänä lineaarisiin sekalukutehtäviin sekä epälineaarisiin tehtäviin. Epälineaarisiin tehtäviin menetelmää on myöhemmin kehitetty. Hajota ja rajoita on implisiittinen luettelointimenetelmä, jossa huonoimmat kokonaislukumuuttujat tiputetaan pois testaamatta niitä. Menetelmässä lähdetään liikkeelle ratkaisemalla alkuperäinen tehtävä relaksoituna lineaarisena tehtävänä. Alussa ja välivaiheissa käytetään lineaarista optimointimenetelmää. (Rao 2009) Nämä tehtävät ratkaistaan usein Simplex-menetelmällä ja niiden ratkaiseminen on yleensä hajota ja rajoita -menetelmässä työläin vaihe (Nocedal & Wright 1999).

Laskentatyökalun diskreettejä tehtäviä varten työssä laadittiin matemaattinen malli, joka voitiin sitten implementoida laskentakoodiin. Diskreettejä joukkoja tarvitaan käytännön suunnittelutehtävissä, joissa muuttujat voivat saada vain tiettyjä diskreettejä re-

aalilukuarvoja. Mikäli muuttujat ovat kokonaislukuja joltain tietyltä väliltä, niin kyseessä on tavallinen kokonaislukutehtävä, jolle annetaan vain ala- ja ylärajat.

Diskreetti muuttuja voidaan valita äärellisestä joukosta $\{s_1,...,s_m\}$ seuraavasti.

$$\begin{gathered} x = \sum_{i=1}^{m} u_i s_i, \\ u_i \in \{0,1\}, j = 1, \dots, m \\ \sum_{i=1}^{m} u_i , = 1, \end{gathered} \tag{26}$$

jossa binäärimuuttujan u_i avulla sallitaan vain yksi diskreetti arvo s_i. Tätä soveltamalla laadittiin seuraavaksi esitetyt diskreetit mallit. Kirjallisuudesta ei löytynyt suoraan käyttökelpoista mallia, jota olisi tässä voitu hyödyntää.

Diskreettejä malleja on laadittu kolmea tapausta varten. Ensimmäinen esitetty diskreetti malli on optimointitehtävälle, jossa kaikki suunnittelumuuttujat valitaan diskreeteistä äärellisistä joukoista. Toinen diskreetti malli on tehtäville, joissa valitaan muuttujat samoin kuin edellisessä, mutta diskreetit muuttujat riippuvat toisistaan. Toisin sanoen, tehtävässä määritellään kaikki sallitut suunnittelumuuttujien kombinaatiot. Käytännössä tämä tarkoittaa usein sallittuja kokoonpanoja. Kolmas diskreetti malli on sekalukutehtäville, joissa voi olla suunnittelumuuttujia kahteen edelliseen tapaukseen liittyen sekä lisäksi jatkuvia muuttujia ja kokonaislukumuuttujia.

Diskreetit mallit on toteutettu lineaarisella yhtälörajoitteella, jonka kerroinmatriisi **A** on täydennetty matriisi, johon lisätään diagonaalimatriisi suunnittelumuuttujia varten sekä diskreeteistä joukoista muodostettu matriisi ja tarvittavat binääriset päätösmuuttujat. Diagonaalimatriisin diagonaalille tulee kerroin -1 jos kyseessä on diskreetti muuttuja. Muussa tapauksessa siihen tulee 0. Täydennetyn matriisiin loppuun voidaan tarvittaessa tavalliseen tapaan lisätä tavallisia yhtälörajoituksia. Tällöin tulee tietysti huomioida, että annetut rajoitukset eivät ole ristiriidassa keskenään. Muuten voi käydä niin, että käypä joukko on tyhjä.

Optimointitehtävään lisätään tavalliseen tapaan vielä kohdefunktio ja mahdolliset muut rajoitusyhtälöt sekä muuttujien ala- ja ylärajat. Vastepintamenetelmää käytettäessä lisätään lineaariset epäyhtälörajoitukset tai kvadraattiset epäyhtälörajoitukset. Optimointitehtävään lisätään muuttujia muodostetun yhtälörajoitteen tarvitsemien päätösmuuttujien verran. Mallit esitetään matriisimuodossa esitystavan selkeyden vuoksi sekä käytännön sovellutusta ajatellen. Mallit on helppo muodostaa esimerkiksi MATLABilla, kuten laskentatyökalussa on tehty.

3.4.1 Diskreetti malli riippumattomille muuttujille

Tehtävä sisältää pelkästään riippumattomia diskreettejä muuttujia. Tarvittava yhtälörajoitus muodostetaan kaavojen (27) ja (28) esittämällä tavalla.

$$\begin{gathered}\mathbf{Ax} = \mathbf{b} \\ \mathbf{A} \in \mathbb{R}^{(2n)x(n+L)} \\ \mathbf{b} = \begin{bmatrix}\mathbf{0}_{nx1} \\ \mathbf{1}_{nx1}\end{bmatrix} \\ x_i \in \mathbf{s}_i \in \mathbb{R}^{1xl_i}, i = 1, \dots, n \\ x_{n+1,\dots,L} \in \mathbb{B} \\ L = \sum_{i=1}^{n} l_i,\end{gathered} \tag{27}$$

missä n on muuttujien lukumäärä,
$\mathbf{s}_i$ on diskreetti joukko muuttujalle x_i,
l_i on diskreettien arvojen lukumäärä vektorissa $\mathbf{s}_i$ ja
L on diskreettien arvojen lukumäärä yhteensä.

Yhtälörajoite

$$\begin{gathered}\mathbf{A} = \begin{bmatrix}-\mathbf{I}_n & \mathbf{B} \\ \mathbf{O}_n & \mathbf{C}\end{bmatrix} \\ \mathbf{B} \in \mathbb{R}^{nxL} \\ \mathbf{C} \in \mathbb{B}^{nxL},\end{gathered} \tag{28}$$

missä

$$\mathbf{B} = \begin{bmatrix}\mathbf{s}_1 & \mathbf{0}_2 & \mathbf{0}_3 & \cdots & \mathbf{0}_n \\ \mathbf{0}_1 & \mathbf{s}_2 & \mathbf{0}_3 & \cdots & \mathbf{0}_n \\ \mathbf{0}_1 & \mathbf{0}_2 & \mathbf{s}_3 & \cdots & \mathbf{0}_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_1 & \mathbf{0}_2 & \mathbf{0}_3 & \cdots & \mathbf{s}_n\end{bmatrix} \tag{29}$$

ja

$$\mathbf{C} = \begin{bmatrix}\mathbf{1}_1 & \mathbf{0}_2 & \mathbf{0}_3 & \cdots & \mathbf{0}_n \\ \mathbf{0}_1 & \mathbf{1}_2 & \mathbf{0}_3 & \cdots & \mathbf{0}_n \\ \mathbf{0}_1 & \mathbf{0}_2 & \mathbf{1}_3 & \cdots & \mathbf{0}_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_1 & \mathbf{0}_2 & \mathbf{0}_3 & \cdots & \mathbf{1}_n\end{bmatrix}, \tag{30}$$

joissa $\mathbf{I}_n$ on identiteettimatriisi,
$\mathbf{O}_n$ on nxn nollamatriisi,
$\mathbf{B}$ on diskreeteistä joukoista $\mathbf{s}_i$ muodostettu matriisi,
$\mathbf{C}$ on valintamatriisi diskreeteille arvoille,
$\mathbf{0}_i$ on $1xl_i$ 0-vektori, $i = 1, \dots, n$ ja
$\mathbf{1}_i$ on $1xl_i$ 1-vektori, $i = 1, \dots, n$.

Avataan ratkaisua seuraavan yksinkertaisen esimerkin kautta.

Esimerkki 2

Annetaan ensin muuttujille diskreetit joukot $\mathbf{s}_i$.
$x_1 \in \mathbf{s}_1 = [7 \quad 9 \quad 13 \quad 16]$,
$x_2 \in \mathbf{s}_2 = [2 \quad 4 \quad 6]$ ja
$x_3 \in \mathbf{s}_3 = [100 \quad 200 \quad 300]$,
jolloin muuttujien lukumäärä $n = 3$ ja
diskreettien arvojen lukumääräksi tulee siten
$l_1 = 4$, $l_2 = 3$ ja $l_3 = 3$.
Diskreettien arvojen lukumäärä yhteensä
$L = l_1 + l_2 + l_3 = 10$.
Päätösmuuttujia tarvitaan lukumäärän L verran, jolloin $x_{4\ldots13} \in \mathbb{B}$.

Sijoitetaan diskreetit joukot $\mathbf{s}_i$ täydennettyyn matriisiin **B** ja laitetaan muut alkiot nolliksi alla kuvatulla tavalla. Diskreeteistä joukoista $\mathbf{s}_i$ muodostettu matriisi

$$\mathbf{B} = \begin{bmatrix} \mathbf{s}_1 & \mathbf{0}_2 & \mathbf{0}_3 \\ \mathbf{0}_1 & \mathbf{s}_2 & \mathbf{0}_3 \\ \mathbf{0}_1 & \mathbf{0}_2 & \mathbf{s}_3 \end{bmatrix} = \begin{bmatrix} 7 & 9 & 13 & 16 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 4 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 20 & 30 \end{bmatrix}.$$

Seuraavaksi muodostetaan täydennetty valintamatriisi **C**, joka on samankokoinen kuin matriisi **B**. Valintamatriisi saadaan kun laitetaan **B**-matriisin diskreettien arvojen tilalle 1. Valintamatriisi

$$\mathbf{C} = \begin{bmatrix} \mathbf{1}_1 & \mathbf{0}_2 & \mathbf{0}_3 \\ \mathbf{0}_1 & \mathbf{1}_2 & \mathbf{0}_3 \\ \mathbf{0}_1 & \mathbf{0}_2 & \mathbf{1}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Seuraavaksi muodostetaan diskreetin mallin yhtälörajoituksen täydennetty kerroinmatriisi **A** siten, että sen vasempaan yläkulmaan tulee ensin nxn identiteettimatriisi kerrottuna -1:llä. Identiteettimatriisin oikealle puolelle sijoitetaan diskreeteistä arvoista muodostettu matriisi **B**, jonka alapuolelle tulee valintamatriisi **C**. Lopuksi lisätään vielä nollamatriisi negatiivisen identiteettimatriisin alle, jotta saadaan täydennetty kerroinmatriisi **A** täydeksi, jolloin

$$\mathbf{A} = \begin{bmatrix} -\mathbf{I}_n & \mathbf{B} \\ \mathbf{O}_n & \mathbf{C} \end{bmatrix}$$

$$= \begin{bmatrix} -1 & 0 & 0 & 7 & 9 & 13 & 16 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 2 & 4 & 6 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 20 & 30 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Lopuksi tarvitaan vielä yhtälörajoituksen oikean puolen vektori **b**, joka saadaan laittamalla siihen nollat identiteettimatriisin rivien kohdalle ja ykköset valintamatriisin osoittamalle kohdalle. Tässä esimerkissä oli vain diskreettejä muuttujia, joten

$$\mathbf{b} = \begin{bmatrix} \mathbf{0}_{nx1} \\ \mathbf{1}_{nx1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Muodostetusta yhtälörajoitteesta $\mathbf{Ax} = \mathbf{b}$ huomataan, että negatiivinen identiteettimatriisi yhdessä valintamatriisin kanssa pakottavat valitsemaan kustakin diskreetistä joukosta vain yhden arvon, jotta ei rikota oikean puolen **b**-vektoria.

3.4.2 Diskreetti malli riippuville muuttujille

Tehtävä sisältää pelkästään toisistaan riippuvia diskreettejä muuttujia. Tarvittava yhtälörajoitus muodostetaan kaavojen (31) ja (32) esittämällä tavalla.

$$\begin{gathered} \mathbf{Ax} = \mathbf{b} \\ \mathbf{A} \in \mathbb{R}^{(n+1)x(n+P)} \\ \mathbf{b} = \begin{bmatrix} \mathbf{0}_{nx1} \\ 1 \end{bmatrix} \\ x_i \in \mathbf{s}_i \in \mathbb{R}^{1xl_i}, \qquad i = 1 \dots n \\ x_{n+1,\dots,P} \in \mathbb{B}, \end{gathered} \tag{31}$$

missä
n on muuttujien lukumäärä,
$\mathbf{s}_i$ on diskreetti joukko faktorille x_i,
l_i on diskreettien arvojen lukumäärä vektorissa $\boldsymbol{s}_i$ ja
P on riippuvien diskreettien joukkojen lukumäärä.

Yhtälörajoite

$$\begin{gathered} \mathbf{A} = \begin{bmatrix} -\mathbf{I}_n & \mathbf{D} \\ \mathbf{0}_{1xn} & \mathbf{1}_{1xP} \end{bmatrix}, \\ \mathbf{D} \in \mathbb{R}^{nxP} \end{gathered} \tag{32}$$

jossa **D**-matriisi muodostetaan riippuvista joukkovektoreista

$$\mathbf{D} = [\mathbf{d}_1 \quad \mathbf{d}_2 \quad \cdots \quad \mathbf{d}_P]. \tag{33}$$

Käydään jälleen ratkaisua läpi hyvin yksinkertaisen esimerkin kautta.

Esimerkki 3

Annetaan ensin muuttujille diskreetit joukot $\mathbf{s}_i$.
$x_1 \in \mathbf{s}_1 = [5 \quad 7 \quad 8 \quad 9]$ ja
$x_2 \in \mathbf{s}_2 = [0.1 \quad 0.2 \quad 0.3 \quad 0.4]$,
joten muuttujien lukumäärä $n = 2$.

Muodostetaan riippuvuusvektorit $\mathbf{d}_i$, joihin tulee näiden kahden muuttujan sallitut kombinaatiot. Riippuvuuksiksi valitaan selkeyden vuoksi vain kolme yhdistelmää. Riippuvuusvektoreiksi saadaan

$$\mathbf{d}_1 = \begin{bmatrix} s_{1,1} \\ s_{2,2} \end{bmatrix} = \begin{bmatrix} 5 \\ 0.2 \end{bmatrix}, \mathbf{d}_2 = \begin{bmatrix} s_{1,2} \\ s_{2,1} \end{bmatrix} = \begin{bmatrix} 7 \\ 0.1 \end{bmatrix} \text{ ja } \mathbf{d}_3 = \begin{bmatrix} s_{1,3} \\ s_{2,4} \end{bmatrix} = \begin{bmatrix} 8 \\ 0.4 \end{bmatrix}.$$

Riippuvien joukkojen lukumäärä $P = 3$, jonka verran tarvitaan päätösmuuttujia, jolloin $x_{3\ldots5} \in \mathbb{B}$.

Muodostetaan riippuvuusvektoreista $\mathbf{d}_i$matriisi

$$\mathbf{D} = [\mathbf{d}_1 \quad \mathbf{d}_2 \quad \mathbf{d}_3] = \begin{bmatrix} 5 & 7 & 8 \\ 0.2 & 0.1 & 0.4 \end{bmatrix}.$$

Seuraavaksi muodostetaan diskreetin mallin yhtälörajoituksen täydennetty kerroinmatriisi **A** siten, että sen vasempaan yläkulmaan tulee ensin nxn identiteettimatriisi kerrottuna -1:llä. Eli vastaavasti kuin edellisessä tapauksessa. Identiteettimatriisin oikealle puolelle sijoitetaan tässä riippuvuusvektoreista muodostettu matriisi **D**, jonka alapuolelle tulee rivi ykkösiä. Lopuksi lisätään vielä nollat negatiivisen identiteettimatriisin alle, jotta saadaan täydennetty kerroinmatriisi **A** täydeksi, jolloin

$$\mathbf{A} = \begin{bmatrix} -\mathbf{I}_n & \mathbf{D} \\ \mathbf{0}_{1xn} & \mathbf{1}_{1xP} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 5 & 7 & 8 \\ 0 & -1 & 0.2 & 0.1 & 0.4 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Lopuksi tarvitaan vielä yhtälörajoituksen oikean puolen vektori **b**, joka saadaan laittamalla siihen nollat identiteettimatriisin rivien kohdalle ja 1 vektorin loppuun. Tässä esimerkissä oli vain diskreettejä muuttujia, joten

$$\mathbf{b} = \begin{bmatrix} \mathbf{0}_{nx1} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Muodostetusta yhtälörajoitteesta huomataan, että negatiivinen iden titeettimatriisi yhdessä joukkojen valintaan liittyvän ykkösrivin kanssa pakottavat valitsemaan vain yhden diskreetin toisistaan riippuvan joukon $\mathbf{d}_i$, jotta ei rikota oikean puolen $\mathbf{b}$-vektoria.

Lopuksi yleistetään edellä esitetyt mallit siten, että optimointitehtävässä voi olla kaikentyyppisiä reaalimuuttujia.

3.4.3 Diskreetti malli sekalukutehtäville

Tehtävä sisältää riippumattomia ja riippuvia diskreettejä muuttujia sekä muita muuttujia. Viimeinen tapaus saadaan soveltamalla edellisiä tapauksia.

$$\begin{gathered}\mathbf{Ax} = \mathbf{b} \\ \mathbf{A} \in \mathbb{R}^{(n+K+1)x(n+L+P)} \\ \mathbf{b} = \begin{bmatrix} \mathbf{0}_{nx1} \\ \mathbf{1}_{Kx1} \\ 1 \end{bmatrix} \\ x_i \in \mathbf{s}_i \in \mathbb{R}^{1xl_i} \text{ tai } x_i \in \mathbb{R}, i = 1, \dots, n \\ x_{n+1,\dots,L+P} \in \mathbb{B},\end{gathered} \tag{34}$$

missä
n on faktoreiden lukumäärä,
$\mathbf{s}_i$ on diskreetti joukko faktorille x_i,
l_i on diskreettien arvojen lukumäärä vektorissa $\boldsymbol{s}_i$,
L on riippumattomien joukkojen diskreettien arvojen lukumäärä yhteensä,
P on riippuvien joukkojen lukumäärä ja
K on riippumattomien joukkojen lukumäärä.

Yhtälörajoite muodostetaan yhdistämällä kaavat (28) ja (32)

$$\begin{gathered}\mathbf{A} = \begin{bmatrix} \mathbf{R}_n & \mathbf{B} & \mathbf{D} \\ \mathbf{O}_{Kxn} & \mathbf{C} & \mathbf{O}_{KxP} \\ \mathbf{0}_{1xn} & \mathbf{0}_{1xL} & \mathbf{1}_{1xP} \end{bmatrix} \\ \mathbf{B} \in \mathbb{R}^{nxL} \\ \mathbf{C} \in \mathbb{B}^{KxL} \\ \mathbf{D} \in \mathbb{R}^{nxP} \\ r_{i,j} = 0, kun\ i \neq j \\ r_{i,j} = -1, kun\ i = j\ ja\ x_i \in \mathbf{s}_i, muuten\ r_{i,j} = 0 \\ d_{i,j} = 0, kun\ x_i \notin \mathbf{d}_i \\ i, j = 1, \dots, n.\end{gathered} \tag{35}$$

Yksinkertaisen esimerkin kautta avataan jälleen ratkaisua.

Esimerkki 4

Tässä esimerkissä on 2 jatkuvaa muuttujaa, 3 toisistaan riippumatonta diskreettiä muuttujaa ja 2 toisistaan riippuvaa diskreettiä muuttujaa.

Annetaan tehtävä kuten edellä.
$x_1, x_5 \in \mathbb{R}$,
$x_2 \in \mathbf{s}_1 = [3 \quad 6]$,
$x_3 \in \mathbf{s}_2 = [2 \quad 4 \quad 8]$,
$x_4 \in \mathbf{s}_3 = [1 \quad 9]$,
$x_6 \in \mathbf{s}_4 = [5 \quad 7 \quad 8]$,
$x_7 \in \mathbf{s}_5 = [0.1 \quad 0.2 \quad 0.4]$,
$n = 7$,
$l_1 = 2$, $l_2 = 3$, $l_3 = 2$ ja
$L = l_1 + l_2 + l_3 = 7$.

Muodostetaan jälleen riippuvuusvektorit $\mathbf{d}_i$, joihin tulee muuttujien x_6 ja x_7 sallitut kombinaatiot. Riippuvuuksiksi valitaan selkeyden vuoksi vain kolme yhdistelmää. Riippuvuusvektoreiksi saadaan

$$\mathbf{d}_1 = \begin{bmatrix} s_{6,1} \\ s_{7,2} \end{bmatrix} = \begin{bmatrix} 5 \\ 0.2 \end{bmatrix}, \mathbf{d}_2 = \begin{bmatrix} s_{6,2} \\ s_{7,1} \end{bmatrix} = \begin{bmatrix} 7 \\ 0.1 \end{bmatrix}, \mathbf{d}_3 = \begin{bmatrix} s_{6,3} \\ s_{7,3} \end{bmatrix} = \begin{bmatrix} 8 \\ 0.4 \end{bmatrix}.$$

Riippuvien joukkojen lukumäärä $P = 3$ ja koska *L*=7, niin tarvitaan 10 päätösmuuttujaa, jolloin $x_{8\ldots17} \in \mathbb{B}$.

Muodostetaan riippuvuusvektoreista $\mathbf{d}_i$vielä matriisi

$$\mathbf{D} = [\mathbf{d}_1 \quad \mathbf{d}_2 \quad \mathbf{d}_3] = \begin{bmatrix} 5 & 7 & 8 \\ 0.2 & 0.1 & 0.4 \end{bmatrix}.$$

Edellisiä diskreettejä malleja soveltamalla muodostetaan diskreetin mallin yhtälörajoituksen täydennetty kerroinmatriisi **A** siten, että sen vasempaan yläkulmaan tulee ensin negatiivinen nxn identiteettimatriisi, jonka diagonaalille tulee tässä nolla, jos kyseinen muuttuja ei ole diskreetti. Identiteettimatriisin oikealle puolelle sijoitetaan diskreeteistä arvoista muodostettu matriisi **B** ja riippuvuusvektoreista muodostettu matriisi **D.** **B**-matriisin alapuolelle tulee valintamatriisi **C**, jonka molemmin puolen tulee nollamatriisit, jotta täydennetty matriisi **A** saadaan täydeksi. Lopuksi lisätään matriisiin vielä rivi riippuvien joukkojen valintaa varten, siten että **D**-matriisin sarakkeiden kohdalle tulee ykköset ja muihin sarakkeisiin tulee nollia. Tällöin

$$\mathbf{A} = \begin{bmatrix} \mathbf{R}_n & \mathbf{B}_{nxL} & \mathbf{D}_{nxP} \\ \mathbf{O}_{Kxn} & \mathbf{C}_{KxL} & \mathbf{O}_{KxP} \\ \mathbf{0}_{1xn} & \mathbf{0}_{1xL} & \mathbf{1}_{1xP} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 3 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 & 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 7 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.1 & 0.4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Lopuksi muodostetaan vielä yhtälörajoituksen oikean puolen vektori **b**, joka saadaan laittamalla siihen nollat identiteettimatriisin rivien kohdalle ja ykköset valintamatriisin osoittamalle kohdalle. Lopuksi lisätään vielä 1 diskreetin joukon valintaa varten, jolloin saadaan

$$\mathbf{b} = \begin{bmatrix} \mathbf{0}_{nx1} \\ \mathbf{1}_{Kx1} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Luvuissa 5.2.5-5.2.6 on esitelty testituloksia tehtäville, joissa on riippuvia ja riippumattomia diskreettejä joukkoja.

3.5 Valmisohjelmat kantavien rakenteiden optimointiin

Seuraavaksi esitellään lyhyesti valmisohjelmia, jotka sopivat kantavien rakenteiden optimointiin ja joilla voidaan tehdä myös monitavoiteoptimointia, jolle usein on tarve. Valintakriteerinä valmisohjelmistolle on myös se, että siinä on valmis tai helposti lisättävissä oleva rajapinta tärkeimpiin lujuuslaskennan suunnitteluohjelmiin, joita ovat esimerkiksi elementtimenetelmäohjelmistot ANSYS ja Abaqus sekä CAD-ohjelmat SolidWorks ja Pro/Engineer. Hajautetun laskennan yhdistäminen ohjelmistoon on myös tärkeä ominaisuus, mutta sen mahdollisuutta ei tarkemmin selvitetty.

Kirjassa Branke et al. (2008) on kerrottu monitavoiteoptimointiin sopivista valmisohjelmista kattavasti. Kirja on tosin julkaistu jo vuonna 2008, jonka jälkeen ohjelmia on varmasti vielä kehitetty ja uusia ohjelmia on tullut markkinoille.

OPTIMUS on kaupallinen laaja ohjelmisto, jossa on kattava kokoelma koesuunnitteluun ja optimointiin liittyviä menetelmiä. Kirjan Branke et al. (2008) kirjoittamisen aikaan OPTIMUS oli maailman johtavia valmisohjelmia tähän tarkoitukseen. OPTIMUS-ohjelmistossa on useita menetelmiä koesuunnitteluun, yhden kohdefunktion optimointiin ja monitavoiteoptimointiin. Ohjelmistossa on käytössä myös erilaisia vastepintamenetelmiä. Käyttäjällä on mahdollisuus lisätä myös omia algoritmeja ohjelmistoon. OPTIMUS tukee myös rinnakkaislaskentaa ja algoritmien ketjuttamista. Rajapintoja muihin ohjelmistoihin on kattavasti, ainakin kaikkiin keskeisiin suunnitteluohjelmistoihin. OPTIMUS-ohjelmistossa löytyy myös työkalu visualisointiin ja tilastolliseen analysointiin. (Branke et al. 2008; OPTIMUS 2014) Ohjelmisto vaikuttaa internet-sivujen mukaan erittäin monipuoliselta (OPTIMUS 2014). Kaksi muuta kattavaa kaupallista ohjelmaa ovat modeFRONTIER ja Isight, jotka tarjoavat kaiken tai lähes saman kuin OPTIMUS (modeFRONTIER 2014; Isigth 2014).

Kaupallisten lisenssien ollessa todennäköisesti erittäin hintavia, on ohjelmistoa tarvitsevan syytä tutustua myös avoimen lähdekoodin ohjelmistoihin. Tässä esitellään lyhyesti vain yhtä hyvin lupaavalta vaikuttavaa ratkaisua DAKOTA. (DAKOTA 2014)

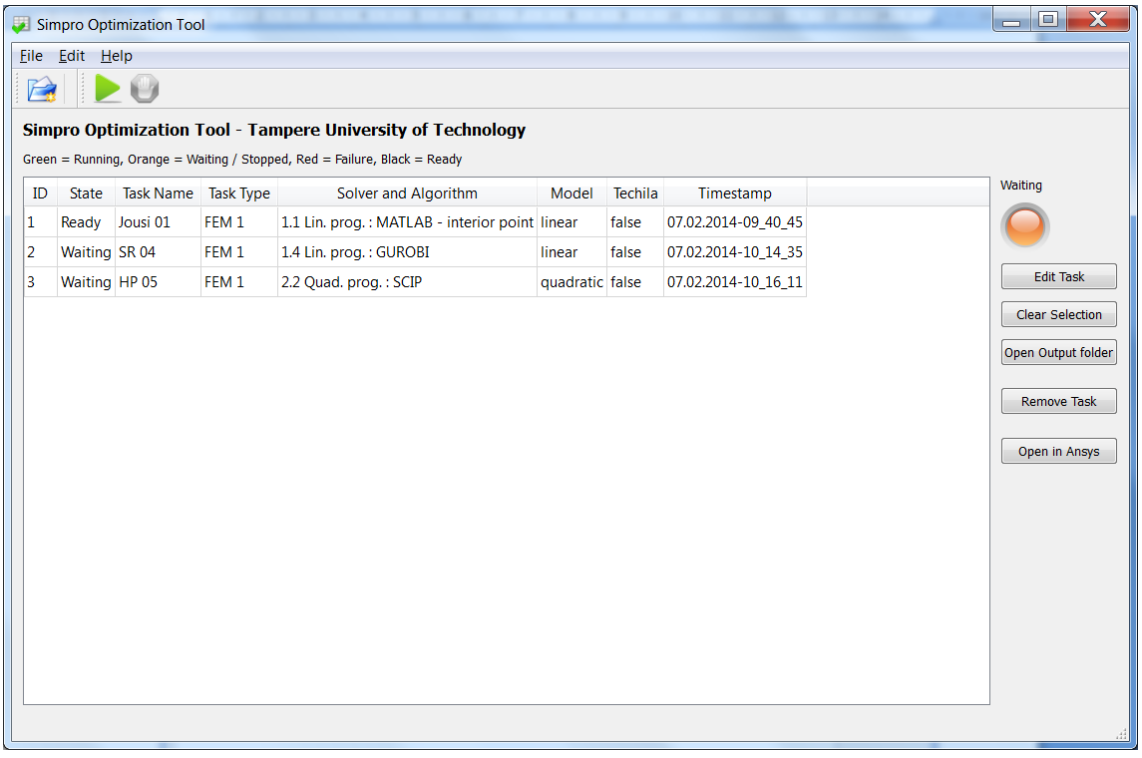
DAKOTA (Design Analysis Kit for Optimization and Terascale Applications) on ohjelmistopaketti, joka sisältää laajan kokoelman muun muassa erilaisia optimointialgoritmeja ja koesuunnittelumenetelmiä. DAKOTA tarjoaa myös valikoiman ratkaisuja rinnakkaislaskentaan. Simulaatiopohjaiseen optimointiin ohjelmistosta löytyy kattava valikoima erilaisia menetelmiä. Graafinen käyttöliittymä JAGUAR GUI on myös erikseen ladattavissa. Olennaisena erona valmisohjelmistoihin on se, että käyttöönotto vaatii huomattavasti enemmän perehtymistä ja myös ohjelmointiosaamista. DAKOTA-ohjelmistoon tulee käyttäjän itse kehittää rajapinta omiin simulointitarpeisiin. DAKOTA-projektin sivuilta löytyy kuitenkin erittäin kattava dokumentaatio. Kehittäjille on perustettu oma portaali, jonne on pääsy myös projektin sivujen kautta. (DAKOTA 2014)

4 LASKENTATYÖKALU

Laskentaympäristö olisi voitu toteuttaa myös käyttämällä esimerkiksi edellisessä luvussa mainittua DAKOTA-ohjelmistoa. Työssä päätettiin kuitenkin laatia oma sovellus, koska siihen liittyvän tutkimuksen kautta saadaan arvokasta tietoa muun muassa sovelluksen kehittämisestä ja käytettävistä menetelmistä. Lisäksi merkittävänä etuna verrattuna valmiiseen alustaan on oman sovelluksen räätälöitävyys.

Tässä luvussa kerrotaan laskentatyökalun toteutuksesta ja käytöstä. Laskentatyökalulle asetetut tavoitteet ja työkalun käyttötarkoitus on kerrottu johdannossa. Viimeisessä alaluvussa kerrotaan työkaluun valittujen algoritmien valintakriteereistä. Lisäksi siinä annetaan suosituksia jatkokehityksessä lisättäviksi algoritmeiksi. Luvussa esitetään myös sovellukseen lisätty taulukko algoritmin valintaan tehtävää varten.

Kuvasta 7 nähdään sovelluksen pääikkuna. Sovelluksen pääikkuna jakautuu neljään osaan, joista hallitsevin on keskellä näkyvä tehtäväjono.



Kuva 7 *Laskentatyökalun pääikkuna.*

Oikeassa reunassa on merkkivalo, josta nähdään selvästi sovelluksen tila. Merkkivalon alapuolella on toimintopainikkeet tehtävän muokkaamiseen, näytön valinnan tyhjentämiseen, tehtävän output-kansion avaamiseen, tehtävän poistamiseen sekä valmistuneen tehtävän parhaimman ratkaisun viemiseen ANSYS-ohjelmaan. Ajon aikana jonossa

olevia tehtäviä ei voida muokata eikä poistaa. Lisäksi uusia tehtäviä ei voida luoda ajon aikana. Muut toiminnot ovat käyttöliittymässä käytössä.

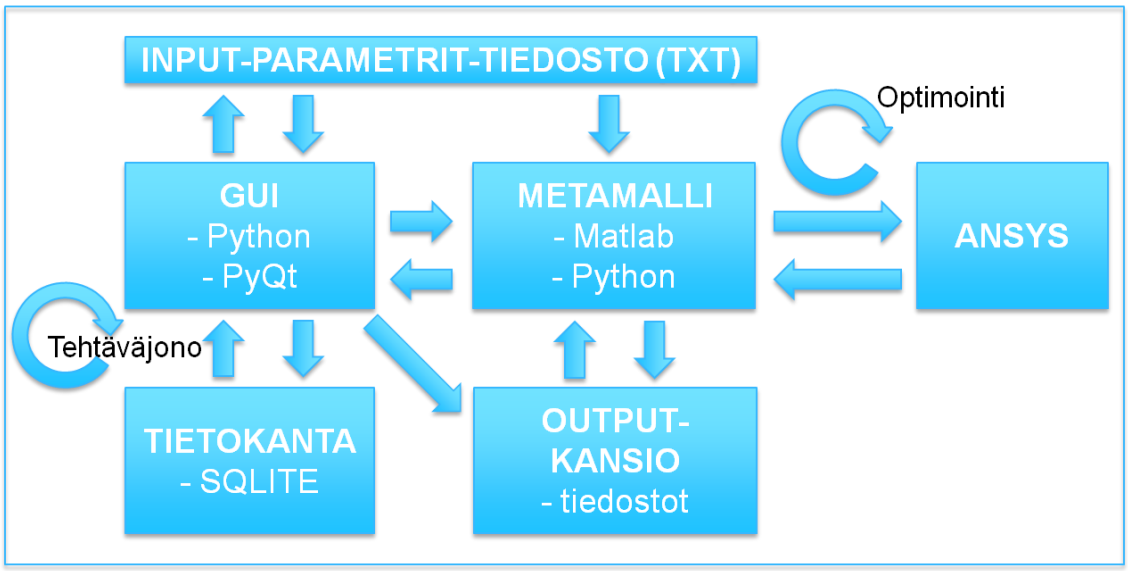
Sovelluksen yläosasta löytyvät työkalurivi ja valikko. Edit-valikon alta löytyvät samat toiminnot kuin painikkeista ja lisäksi toiminto ”Clear all”, jolla voidaan tyhjentää koko tehtäväjono. Lisäksi Edit-valikon alta löytyy pääsy asetuksiin, jossa käyttäjä määrittelee omien ohjelmien sijainnit. Tällä hetkellä asetuksiin määritetään vain MATLAB- ja ANSYS-ohjelman sijainti. Muuta käyttöönottoon liittyvää käyttäjän ei tarvitse tehdä. File-valikon kautta voidaan lisätä uusi tehtävä tai poistua sovelluksesta. Help-valikosta löytyvät perinteiset About- ja Help-toiminnot. Sovelluksen Help-sivut on tehty helposti päivitettäväksi, sillä ne ovat html-muotoiset.

Valikon alapuolelta löytyvältä työkaluriviltä löytyvät toiminnot uuden tehtävän luomiseen, tehtäväjonon ajamiseen sekä ajon pysäyttämiseen. Pysäyttäminen ei pysäytä MATLAB-laskentakoodin suorittamista, vaan se täytyy erikseen pysäyttää. Pysäyttämisen jälkeen tehtävää voidaan muokata ja ajo voidaan käynnistää uudestaan.

Sovellukseen on toteutettu myös pikanäppäimiä, joiden polut näkyvät valikon toimintojen vieressä. Esimerkiksi uusi tehtävä voidaan lisätä näppäinyhdistelmällä Ctrl-N.

4.1 Toteutus ja käyttö

Toteutus on esitetty yleisellä tasolla kuvassa 8. Metamalli-laatikko kuvaa tehtyä laskentakoodia, joka tehtiin MATLABilla. Kuvasta voidaan nähdä, että ratkaisussa on neljä rajapintaa, jotka ovat itse käyttöliittymä (GUI), tietokanta, MATLAB ja ANSYS-ohjelma. Kuvasta nähdään myös näiden vuorovaikutus. Input-parametri-tiedosto on käyttöliittymän tai tekstieditorin kautta muokattava parametritiedosto MATLABille. Uutta tehtävää luodessa, luodaan samalla output-kansio tehtävän parametreille ja ajon jälkeisiä tuloksia varten. Parametrit tallentuvat myös tietokantaan käyttöliittymää varten.



Kuva 8 *Laskentatyökalun toteutus.*

Kuten kuvasta 8 nähdään, niin laskentatyökalun käyttöliittymä ja ANSYS-kytkentä on ohjelmoitu Pythonilla. Tietokanta on toteutettu SQLite-relaatiotietokantajärjestelmällä. Tietokannassa on vain yksi taulu, jossa on kaikki tehtäviin liittyvät parametrit sekä muuta käyttöliittymän tarvitsemaa tietoa. Sovellusta suunnitellessa huomioitiin laajentamistarpeet noudattamalla modulimaista ajattelua ja laskentakoodin osalta laatimalla funktioita, jotka ovat helposti ylläpidettävissä.

Loppukäyttäjiä varten sovellus voidaan kääntää helposti Windowsissa ajettavaksi. Laskentatyökalun käyttö vaatii loppukäyttäjiltä MATLAB- ja ANSYS-ohjelmistot sekä tarvittavat optimointikirjastot.

Työn pohjana ollut laskentakoodi meni melkein kokonaan uusiksi, mutta keskeinen idea siitä tietysti on jäljellä eli vaiheittaisen vastapintamenetelmän käyttäminen optimointiin. Lineaarisen metamallin lisäksi laskentakoodiin lisättiin kvadraattinen ja puhtaasti kvadraattinen metamalli. Lisäksi laskentakoodiin lisättiin muun muassa useita vaihtoehtoisia optimointialgoritmeja, historiadatan kerääminen raportointia varten sekä useita informatiivisia kuvaajia, joista nähdään ajon aikana hyödyllisiä tietoja. Laskentakoodiin implementoitiin myös luvussa 3.4 esitetyt diskreetit mallit.

Liitteestä 3 nähdään laskentatyökalun laskennassa käyttämät MATLAB-tiedostot ja muut tiedostot. Käyttöliittymään liittyviä tiedostoja ei tässä esitellä, koska ne eivät vaikuta itse laskentaan.

Laskennan voi käynnistää myös ilman käyttöliittymää asettamalla parametrit ”input_parameters.txt”-tiedostoon ja ajamalla sitten ”start_optim.bat”-tiedoston. Käyttöliittymän avulla tehtäviä voidaan helposti lisätä tehtäväjonoon sekä hallinnoida niitä. Käyttöliittymä luo jokaiselle tehtävälle oman kansion, johon raportit ja muut tiedostot siirretään ajon lopuksi. Ajettaessa tehtävää ilman käyttöliittymää, jäävät laskentakoodin tuottamat raportit output-kansion juureen, josta ne voidaan ottaa sitten talteen.

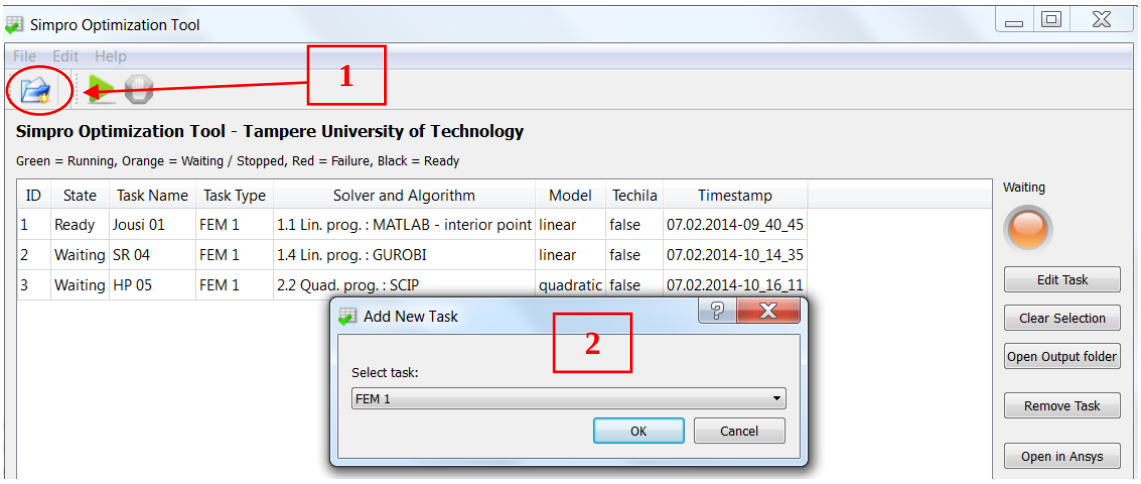
Seuraavaksi esitellään laskentatyökalun toteutus tarkemmin vuokaavioiden ja kuvakaappauksien tukemana. Toteutuksen kuvauksessa ei mennä ohjelmallisiin yksityiskohtiin, esimerkiksi siihen miten sovellus on ohjelmoitu eikä ohjelmointikoodeja esitetä. Toteutuslogiikka kuitenkin esitetään, että miten sovellus on vuorovaikutuksessa laskentakoodin kanssa. Kuten edellä kerrottiin, niin laskentakoodi on oma kokonaisuus ja tehtäviä voidaan ajaa myös ilman käyttöliittymää. Tästä on etuna se, että käyttöliittymä voidaan korvata halutessa toisella käyttöliittymällä, johon vain sitten kytketään toteutettu laskentakoodi.

Seuraavaksi kerrotaan ensin tehtävien lisäämisestä, muokkaamisesta sekä tehtäviin liittyvistä parametreista. Tämän jälkeen kerrotaan tehtävien ajamisesta ja varsinaisen optimoinnin toteutuksesta sekä tulosten raportoinnista.

4.1.1 Tehtävän lisääminen ja muokkaaminen

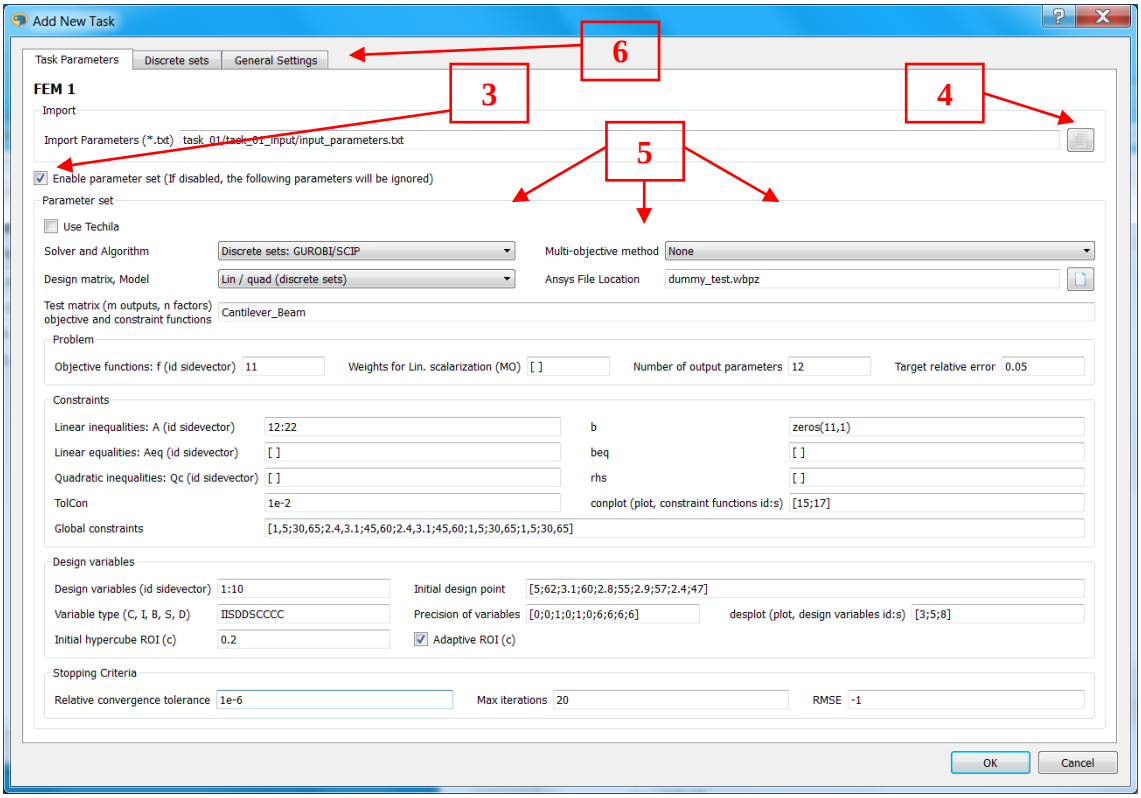
Liitteen 4 vuokaaviosta nähdään tehtävän lisäämiseen liittyvät vaiheet. Tehtävä lisätään klikkaamalla käyttöliittymästä ”Add New Task”-painiketta ja valitsemalla avautuvasta dialogista tehtävätyyppi (Kuva 9: kohdat 1-2). Työkaluun on lisätty tehtävätyyppi

”FEM 1”, jolla optimoidaan käyttämällä vaiheittaista vastepintamenetelmää. Jatkokehityksessä työkaluun on tarkoitus lisätä myös muita tehtävätyyppejä.



Kuva 9 *Tehtävätyypin valinta.*

Avautuneesta ikkunasta valitaan, että tuodaanko tehtävän parametrit erillisestä tekstitiedostosta vai syötetäänkö ne lomakekenttien kautta (Kuva 10: kohdat 3-4). Parametrit tuodaan poistamalla valinta kohdasta ”Enable parameter set” ja hakemalla haluttu tekstitiedosto sekä hyväksymällä tiedot napsauttamalla ”OK”.



Kuva 10 *Tehtävän parametrit 1/3.*

Tehtävän tallennuksessa luodaan MATLABin tarvitsemia parametreja varten ”input.parameters.txt”-tiedosto (Liite 9), jota voidaan hyödyntää myöhemmin uudestaan.

Vastaavantyyppistä tehtävää lisättäessä on hyvä ottaa vanha tehtävä pohjaksi ja muokata sitä käyttöliittymän tai tekstitiedoston kautta. Tämä nopeuttaa tehtävän syöttöä ja vähentää virheensyöttömahdollisuutta.

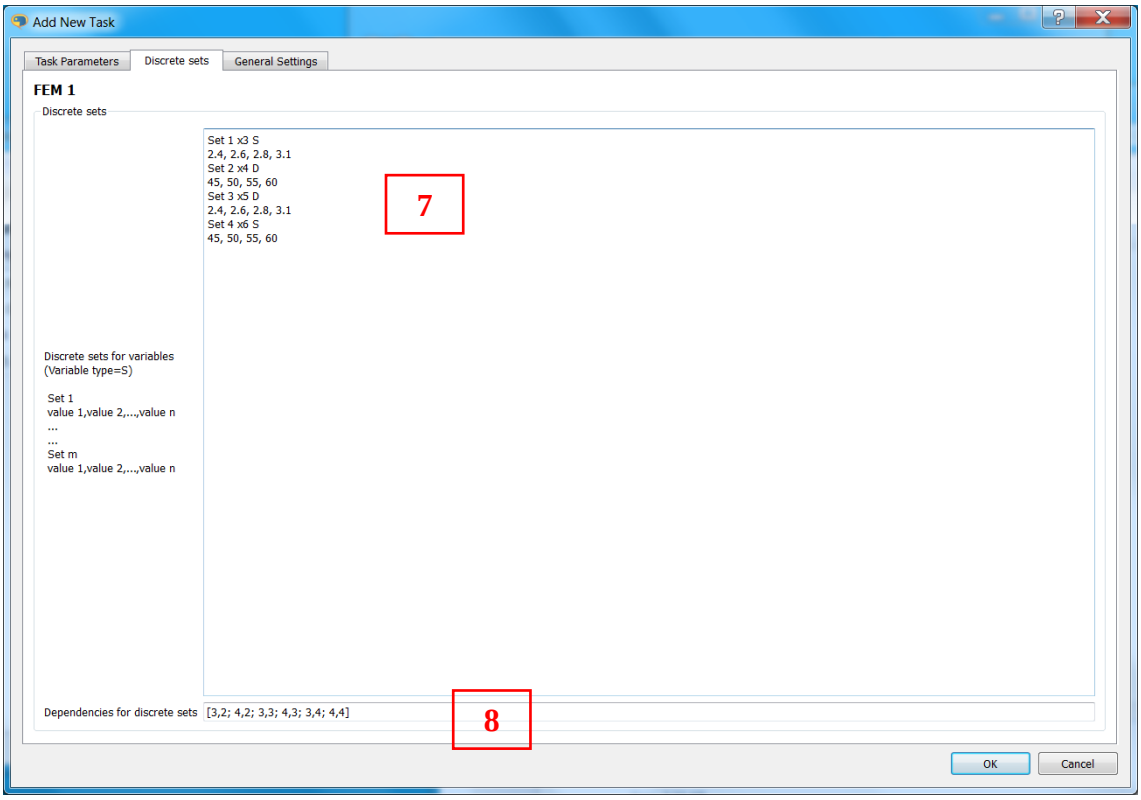
Tekstikenttiin voidaan antaa MATLABin syntaksia. Toistuvat elementit on helppo antaa MATLABin komennoilla. Taulukossa 2 esitetään muutamia hyödyllisiä MATLAB-komentoja tietojen syöttämiseen.

Taulukko 2 *MATLAB-komentoja tiedon syöttämisen helpottamiseksi.*

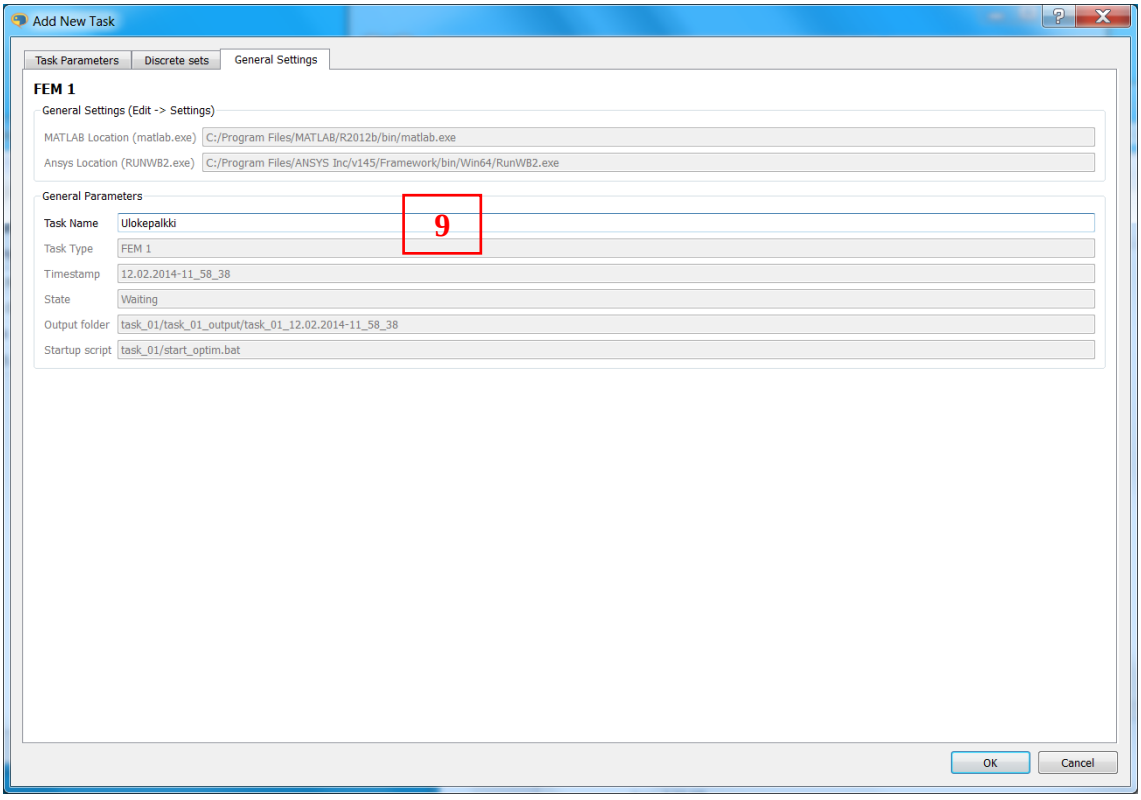
Tavoite	MATLAB-komento
[0; 0; 0; 0; 0; 0]	zeros(6, 1)
[1, 2, 3, 4]	1:4
[3; 3; 3; 3; 3]	3*ones(5, 1)
[2, 4; 2, 4; 2, 4; 2, 4; 2, 4; 2, 4]	repmat([2, 4], 6, 1)
$x_1, \dots, x_5 = [8, 12]$ $x_6, \dots, x_{21} = [15, 20]$	[repmat([8, 12], 5, 1); repmat([15, 20], 16, 1)]

Käyttöliittymässä parametrien syöttö on jaettu kolmelle välilehdelle (Kuva 10: kohta 6). Välilehdet nähdään kuvista 10-12. Parametrit annetaan valikoiden, valintojen ja tekstikenttien avulla (Kuva 10: kohta 5, Kuva 11: kohdat 7-8 ja Kuva 12: kohta 9). Kaikki tehtäviin liittyvät parametrit on esitelty lyhyesti liitteessä 1.

Kaikki lomakekentät ovat ensimmäisellä välilehdellä pakollisia, eikä niissä ole vielä muuta virheentarkistusta kuin pakollisuus. Mikäli jokin tieto on annettu virheellisesti, esimerkiksi rajojen lukumäärä on pienempi kuin faktoreiden lukumäärä, niin paljastuu se vasta laskentakoodia ajettaessa. Laskentakoodia suoritettaessa voi tulla MATLABin omia tai laskentakoodiin tehtyjä virheilmoituksia. Esimerkiksi jos jokin vektori tai matriisi on vääränkokoinen, pysähtyy laskenta yleensä MATLABin virheilmoitukseen. Laskentakoodissa on jossain tilanteissa varauduttu virhetilanteisiin, joille on tehty oma virheidenkäsittely, jolloin laskenta ei pysähdy MATLABin virheilmoitukseen. Esimerkiksi optimointialgoritmeihin on lisätty try-catch -tarkistus, jonka avulla vaihdetaan virheen tullessa algoritmi MATLABin Interior Point -algoritmiin (Matlab 2013). Tästä myös ilmoitetaan käyttäjälle MATLABin komentoikkunassa.



Kuva 11 *Tehtävän parametrit 2/3.*



Kuva 12 *Tehtävän parametrit 3/3.*

Viimeiseltä välilehdeltä (Kuva 12) nähdään myös muita yleisiä parametreja, jotka sovellus lisää automaattisesti eikä niitä voida käsin muuttaa.

Optimointitehtävien asetuksista puhuttaessa pitää erotella työkaluun ja optimointialgoritmeihin liittyvät asetukset, joista kerrotaan luvussa 4.2. Seuraavaan taulukkoon 3 on koottu työkaluun liittyvät keskeisimmät yleiset asetukset, joista kaikkia voidaan muuttaa tehtäväkohtaisesti käyttöliittymän kautta. Taulukkoon on annettu esimerkkinä testiprobleemien ja ANSYS-mallien optimoinnissa käytettyjä arvoja. Taulukosta nähdään myös käytetyt lyhenteet.

Taulukko 3 *Työkaluun liittyvät yleiset asetukset.*

Asetus	Lyhenne	Kuvaus	Arvot
Lopetusehto (konvergenssi)	convtol	Katso kaava (39)	1e-12…1e-3 (1e-10…0.1 %)
Maksimi iteraatiot		Lopetusehto	10…200
ROI:n säde	α	Tarkasteltavan aliavaruuden koko, annetaan ROI:n alkusäde.	0.1, 0.2, tai 0.5 (10, 20 tai 50 %)
Adaptiivinen ROI:n säde		Muuttuva tai kiinteä ROI:n säde.	kyllä / ei
Suunnittelumuuttujien tarkkuus		Annetaan muuttujakohtaisesti vektorina.	1e-12…0
Tavoitevirhe	TolRel	Käytetään uuden ROI:n säteen laskemisessa. Katso kaava (38).	1e-3, 0.01, 0.05 tai 0.1 (0.1, 1, 5 tai 10 %)
Keskiarvo rajoitusyhtälöiden rikkomiselle	$\overline{\boldsymbol{gviol}}$	Katso kaava (40)	
Virhetoleranssi rajoitusyhtälöille	TolCon	Absoluuttinen toleranssi, käypä kun keskiarvo $\overline{\boldsymbol{gviol}} \leq$ TolCon	1e-4…1e-2

Liitteessä 1 on esitelty lyhyesti kaikki tehtäviin liittyvät parametrit. Osaa parametreista on selitetty tarkemmin luvussa 4.1.2, jossa kerrotaan tehtävien ajamisesta ja optimoinnista.

Käyttäjä voi valita, että käytetäänkö ANSYS-tehtävän ajamisessa Techila-laskentaympäristöä vai ajetaanko tehtävä pelkästään paikallisesti omalla tietokoneella. Techilaa ei voida käyttää testifunktioiden kanssa. Techilan ollessa käytössä lähetetään koepisteet määritetyn laskentaympäristön koneille laskettavaksi. Optimoimalla saadun optimipisteen vasteet lasketaan kuitenkin aina paikallisesti. Luvussa 2.3 on kerrottu yleistä Techilasta.

Keskeisin valinta on algoritmin ja mallin valinta, joka vaikuttaa tehtävän ajamisen kestoon ja tarkkuuteen. Luvussa 4.2 on kerrottu tarkemmin työkaluun implementoiduista optimointialgoritmeista ja miten se valitaan tehtävän tyypin mukaan.

Työkaluun on lisätty myös monitavoiteoptimointi, mutta sitä ei tässä käsitellä. Monitavoiteoptimoinnin valintaan valitaan ”None” yhden kohdefunktion tehtäviin.

Optimoitaessa ANSYS-mallia annetaan parametreihin kyseisen mallin tiedostopolku. Techila-laskentaympäristöä käytettäessä ANSYS-mallin tulee olla pakattu eli ”.wbpz”-päätteinen tiedosto. Paikallisesti ajettuna laskentatyökaluun voidaan tuoda

myös pakkaamaton FEM-malli. Laskentatyökalun kehitysvaiheessa mallia ei saatu toimimaan Techilan kanssa, jos mallissa oli SolidWorks-kytkentä. Esimerkiksi Tuloksetluvussa analysoitu dynaaminen palkki oli tällainen. Testifunktioita optimoitaessa ANSYS-tiedoston polkuun on annettava ”dummy.wbpz”, jotta laskentakoodi tietää kyseessä olevan testifunktio.

Testiprobleema annetaan ”Test matrix” -kenttään matriisina tai MATLAB-tiedostona, jossa tehtävä on määritelty. Vasteet tulevat matriisin riveittäin seuraavasti: [(x1-8)^2+(x2-2)^2; 0.1*x1^2-x2; (1/3)*x1+x2-4.5]. Esimerkissä on annettu kolme vastetta. ANSYS-mallia optimoitaessa kenttään annetaan tyhjä matriisi [ ].

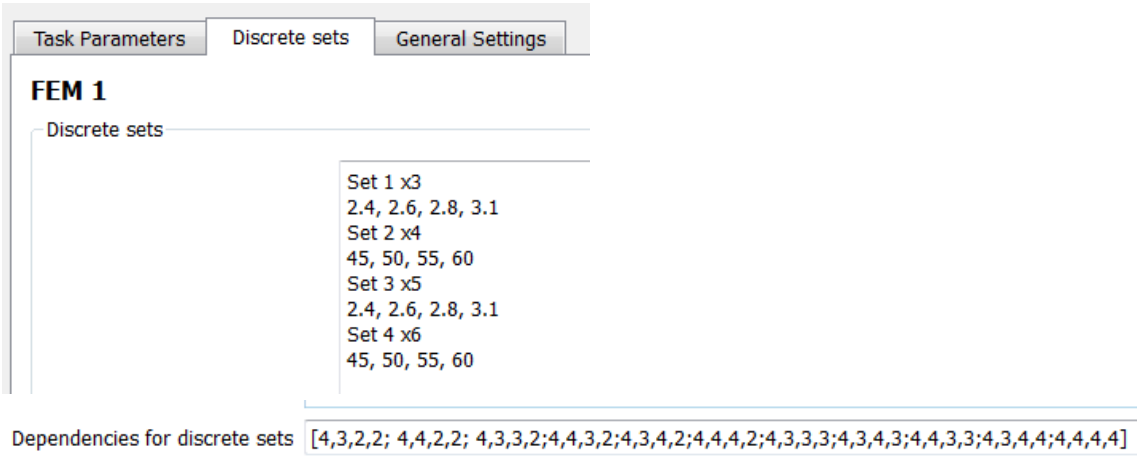
Tehtävän määrittelyssä kohdefunktioihin eli ”Objective functions”-kenttään annetaan minimoitavien vasteiden ID:t vaakavektorina. Rajoitteina toimivien vasteiden ID:t syötetään vastaavasti. ID:t nähdään ANSYS-mallin parametreista. Optimoitaessa testifunktiota ID:t voidaan antaa vapaasti, kuitenkin siten että niiden järjestys vastaa annettua tehtävää. Esimerkiksi yllä mainitulle tehtävälle voisi antaa kohdefunktion ID:ksi 3 ja rajoitefunktioihin [4, 5]. Tämä tarkoittaa sitä, että kohdefunktio on ensimmäinen testimatriisissa ja seuraavat ovat rajoitefunktioita. Rajoitteina toimiville vasteille annetaan vielä ylärajat vektorina.

Tehtävän suunnitteluavaruus annetaan *n*x2-matriisina, jossa on muuttujien ala- ja ylärajat. Muuttujien ID:t annetaan vastaavasti kuin vasteet. Suunnittelumuuttujien tarkkuus annetaan vektorina ilman etumerkkiä, esimerkiksi [2; 4] tarkoittaa samaa kuin [1e-2, 1e-4]. Mikäli jokaisen muuttujan tarkkuus on sama, niin voidaan antaa pelkästään yksi luku. Tehtävälle täytyy antaa myös aloituspiste vektorina, joka otetaan ensimmäisen iteraation aliavaruuden keskipisteeksi. ROI:n alkusäteen avulla saadaan ala- ja ylärajat ensimmäiselle aliavaruudelle. Tehtävälle voidaan asettaa käyttöön myös adaptiivinen ROI:n säde, jolloin aliavaruuden kokoa kasvatetaan iteraatioiden aikana jos mallin antamien ja todellisten vasteiden välillä on vain vähän virhettä. Tavoitevirhettä nostamalla ROI:n sädettä kasvatetaan helpommin. Tämä on implementoidun menetelmän keskeinen idea. Siitä on kerrottu tarkemmin luvussa 4.1.2.

Lopetusehtoina laskentatyökalussa käytetään iteraatioiden maksimimäärää ja suhteellista konvergenssia. Tehtävän määrittelyn ensimmäiseltä välilehdeltä valitaan myös iterointien aikana piirrettävät kuvaajat. Kuvaajista ja muusta raportoinnista on kerrottu tarkemmin luvussa 4.1.3.

Toisella välilehdellä annetaan diskreetit joukot muuttujille, joissa arvot valitaan diskreeteistä joukoista. Diskreetit joukot annetaan samassa järjestyksessä kuin ne on annettu ensimmäisen välilehden muuttujatyyppeihin. Diskreettejä muuttujatyyppejä ovat S ja D, joista jälkimmäinen on riippuva muuttujatyyppi. ”Discrete sets for variables” -kenttään annetaan muuttujittain ensin otsikko ja sitten sen alapuolelle diskreetti joukko vektorina. Diskreettien joukkojen syöttämisen jälkeen määritellään riippuville muuttujille vielä riippuvuudet välilehden alimmaiseen kenttään. Riippuvuudet muodostetaan diskreettien joukkojen vektoreiden indeksien avulla. Kuvassa 13 on esimerkki, jossa muuttujat x_3...x_6 ovat riippuvia toisistaan. Esimerkiksi ensimmäinen riippuva joukko muodos-

tetaan indekseistä 4, 3, 3 ja 2, jotka vastaavat diskreettien joukkojen arvoja 3.1, 55, 2.6 ja 50.



Kuva 13 *Diskreetit joukot ja riippuvuudet.*

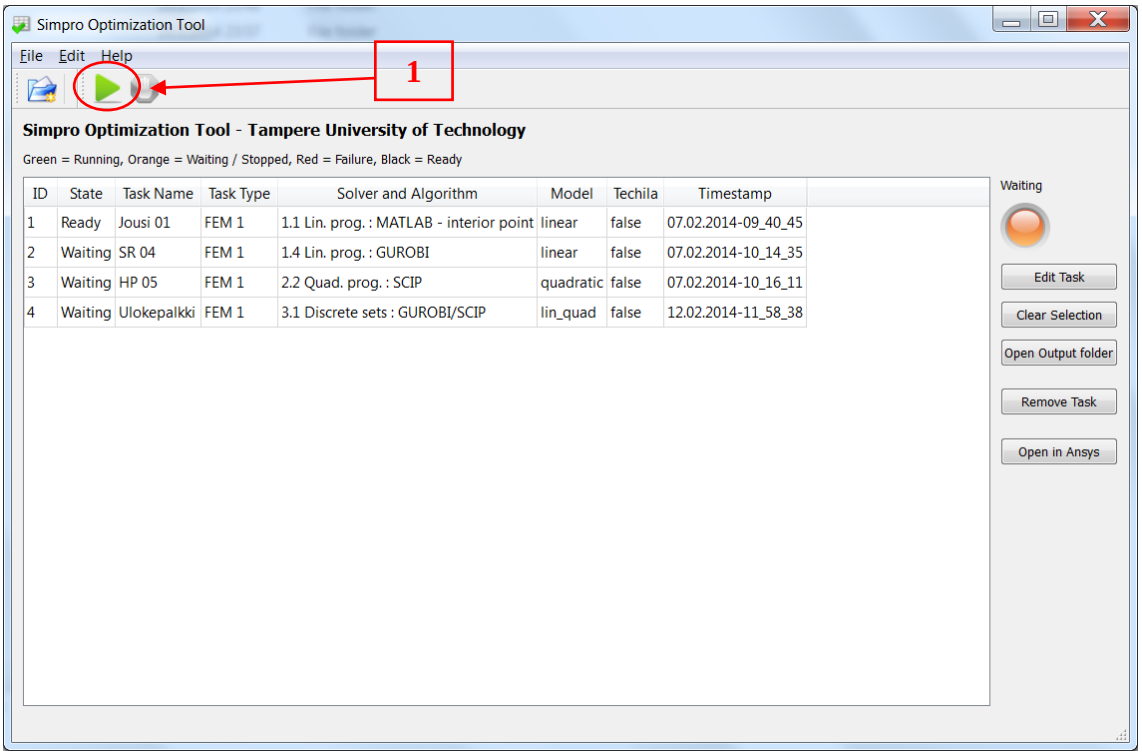
Kun uusi tehtävä on lisätty, tallennetaan tehtävä tietokantaan ja samalla luodaan kansio tuloksia varten. Tulos- eli output-kansioon tulee myös ”input.parameters.txt”-tiedosto. Lisätty tehtävä ilmestyy pääikkunan tehtäväjonoon. Tehtävää voidaan vielä muokata tallennuksen jälkeen kaksoisklikkaamalla tehtävää tai klikkaamalla ”Edit Task”-painiketta. Tehtävää muokatessa päivitetään parametrit tietokantaan ja tekstitiedostoon. Tehtävän muokkaamiseen liittyvät vaiheet on esitetty liitteessä 5.

Sovelluksen pääikkunan tehtäväjonosta nähdään lisättyjen tehtävien tila. Tehtävän lisäyksen jälkeen tilaksi tulee ”Waiting”. Tehtävän ollessa ajossa muuttuu tilaksi ”Running”. Mikäli ajo keskeytetään, muuttuu tilaksi ”Stopped”. Valmiit tehtävät merkitään ”Ready”. Valmiita tehtäviä ei voida enää muokata. Lisäksi ajon aikana odottavia tehtäviä ei voida muokata eikä uusia tehtäviä lisätä. Tästä syystä ajettavat tehtävät on hyvä suunnitella tarkkaan etukäteen. Tehtäviä voidaan myös poistaa kun ajo ei ole käynnissä. Sovelluksen ”Edit”-valikosta löytyy valinta ”Clear all”, jolla voidaan kerralla poistaa kaikki tehtävät tehtäväjonosta. Tehtäväjonosta poistaminen ei kuitenkaan poista output-kansioita, joten tulokset pysyvät tallessa. Käyttäjä voi käsin poistaa output-kansioita tai siirtää ne toiseen hakemistoon talteen.

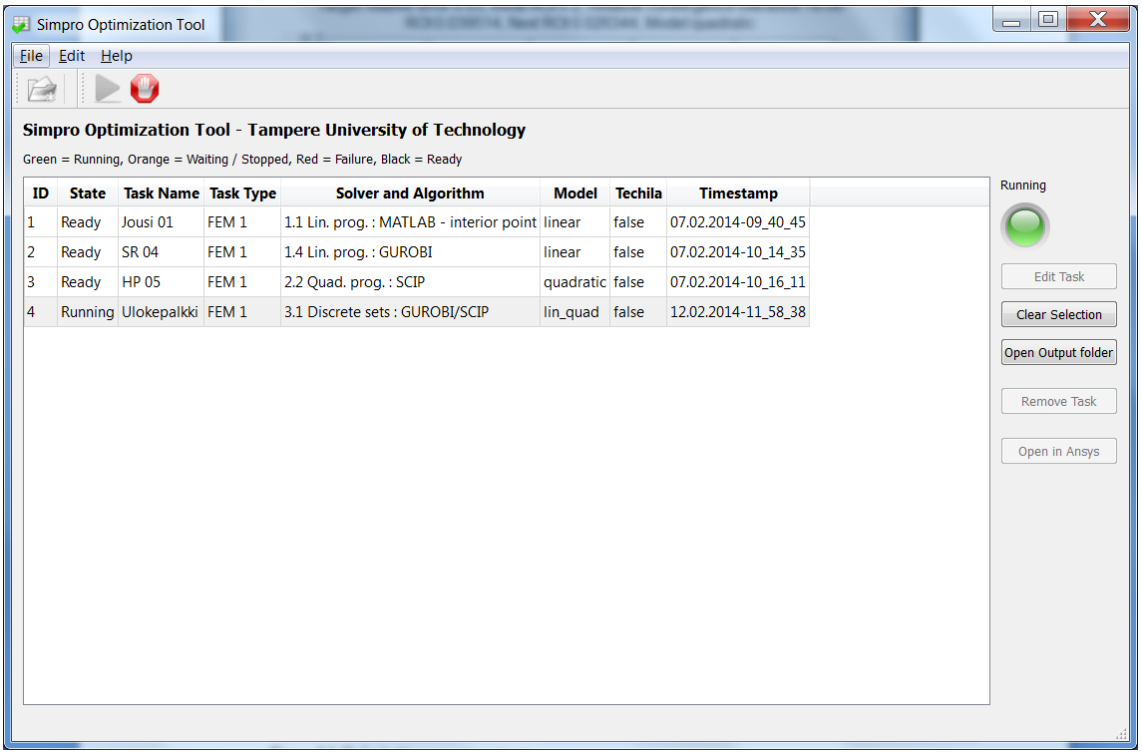
4.1.2 Tehtävien ajaminen

Liitteen 6 vuokaaviosta nähdään tehtävien ajamiseen liittyvät vaiheet. Tehtävien lisäyksen jälkeen voidaan käynnistää tehtäväjonon ajaminen napsauttamalla työkaluriviltä ”Start”-painiketta (Kuva 14: kohta 1). Sovellus hakee tietokannasta seuraavan odottavan tehtävän ajoon ja merkitsee sen tilaksi ”Running”. Lisäksi käyttöliittymästä lukitaan kielletyt ajonaikaiset toiminnot, kuten tehtävien poistaminen. Kuvasta 15 nähdään käyttöliittymän tila ajonaikana. Tehtävään liittyvä parametritiedosto kopioidaan input-kansioon, josta laskentakoodi hakee parametrit. Seuraavaksi sovellus ajaa ”start.bat”-tiedoston, joka käynnistää laskentakoodin suorittamisen MATLABin komentoikkunas-

sa. Vuokaavioissa optimointitehtävän ajaminen on jaettu kahteen alivuokaavioon (Liite 7 ja 8).



Kuva 14 *Tehtäväjono.*



Kuva 15 *Tehtäväjonon ajaminen.*

Liitteen 7 vuokaaviosta nähdään laskentakoodin suorittamat päävaiheet. Laskentakoodissa luodaan ensin log-tiedosto, johon ajon lopuksi tallennetaan MATLABin komentoikkunan tapahtumat. Log-tiedostosta nähdään myös laskennan välivaiheet jotka ovat tarpeen erityisesti vikatilanteissa. Laskentakoodissa kirjoitetaan heti alussa myös ”state.txt”-tiedostoon tilaksi ”Running”. Sovellus tietää tästä tilasta, koska ajo on valmis. Optimointitehtävän valmistuessa suljetaan log-tiedosto ja kirjoitetaan ”state.txt”-tiedostoon tilaksi ”Ready”.

Liitteen 8 vuokaaviosta nähdään varsinaisen optimoinnin vaiheet. Optimointitehtävää varten luetaan sovelluksen input-kansion ”input_parameters.txt”-tiedostosta tehtävän parametrit MATLAB-ohjelman työpöytämuuttujiin. ANSYS-ohjelmaa varten päivitetään tiedostoja ”MatlabAnsys.py” ja ”xbestsolution.py”. Näihin tiedostoihin tulevat ANSYS-ohjelman tarvitsemat vasteiden ja suunnittelumuuttujien ID:t. Techilaa varten kirjoitetaan ”testi.bat”-tiedostoon ANSYS-tiedoston nimi. Alustusten jälkeen alkaa varsinainen laskentaproseduuri. Laskentaproseduuri voidaan jakaa viiteen päävaiheeseen, jotka ovat:

1. Määritetään koepisteet ROI:n sisällä
2. Ajetaan analyysit ANSYS-ohjelmassa
3. Luodaan vastepinnat
4. Lasketaan optimointitehtävä hyödyntäen kohde- ja rajoitusfunktioiden vastepintoja.
5. Tarkastetaan lopetusehto, ja jos ei toteudu niin päivitetään ROI ja jatketaan kohdasta 1. Muuten lopetetaan iterointi.

Päävaiheet on kuvattu myös vuokaavion kohdissa 34.4.5-34.4.12.

Vaiheessa 1 määritetään koepisteet ROI:n sisällä seuraavaksi kuvatulla tavalla. Ensin määritetään ROI:n rajat seuraavalla kaavalla.

$$(1-\alpha_k)\mathbf{x}_{k-1}^* \leq \mathbf{x}_{k-1}^* \leq (1-\alpha_k)\mathbf{x}_{k-1}^*, \tag{36}$$

missä
$\mathbf{x}^* \in \mathbb{R}^n$,
k on iteraatio,
$\mathbf{x}_{k-1}^*$ on edellisen iteraation optimipiste, kun $k \geq 2$,
$\mathbf{x}^* = \mathbf{x}_0$, kun $k = 1$ ja
α_k on ROI:n säde.

Rajat saadaan ottamalla edellisen iteraatiokierroksen optimipiste ROI:n keskipisteeksi ja käyttämällä määritettyä ROI:n sädettä. Ensimmäisellä iteraatiolla käytetään aloituspistettä $\mathbf{x}_0$ ja tehtävässä ennalta annettua aloitusarvoa ROI:n säteelle. Globaalit rajat rajoittavat iteraatiokierroksilla käytettyjä rajoja. Jos uusi raja on lähellä globaalia rajaa, niin siirretään rajoja ylityksen verran toiseen suuntaan. Esimerkiksi jos globaalit rajat ovat

[2, 10] ja aliavaruuden rajoiksi saadaan kaavalla (36) [7.84, 11.76], niin siirtämisen jälkeen saadaan [7.84-(11.76-10) ,10], joka on [6.08, 10]. Vastaavasti toimitaan alarajan kanssa. Näin toimimalla ROI:n koko ei pienene turhaan liikaa. Laskentatyökalussa voidaan käyttää kiinteää tai adaptiivista ROI:n sädettä.

Adaptiivista ROI:n sädettä käytettäessä lasketaan ensin suhteelliset virheet

$$\mathbf{r}_{k-1} = \left| \frac{\hat{y}_i(\mathbf{x}^*_{k-1}) - y_i(\mathbf{x}^*_{k-1})}{y_i(\mathbf{x}^*_{k-1})} \right| \tag{37}$$

missä
$\mathbf{x}^* \in \mathbb{R}^n$,
k on iteraatio,
$\mathbf{x}^*_{k-1}$ on edellisen iteraation optimipiste,
$\hat{y}_i$ on iteraatiolla *k*-1 saadun mallin antamat vasteet,
y_i on iteraatiolla *k*-1 saadun todelliset vasteet,
$i = 1,\ldots,p$ ja
p = vasteiden lukumäärä,

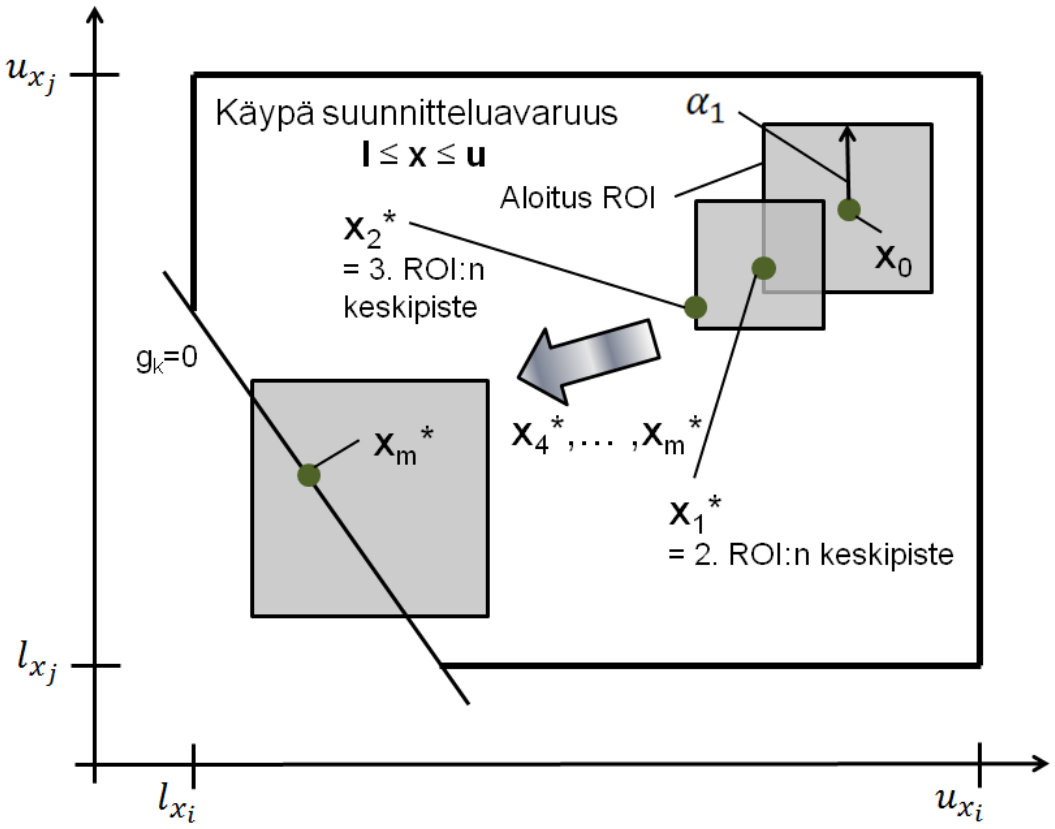
jonka avulla voidaan määrittää ROI:n säde

$$\begin{gathered} \alpha_k = 1.5\alpha_{k-1}, \text{ kun TolRel}/\max(\mathbf{r}_{k-1}) > 1{,}5 \\ \alpha_k = 0.6667\alpha_{k-1}, \text{ kun TolRel}/\max(\mathbf{r}_{k-1}) < 0.6667 \\ \text{muulloin} \\ \alpha_k = (\text{TolRel}/\max(\mathbf{r}_{k-1}))\alpha_{k-1}, \end{gathered} \tag{38}$$

missä TolRel on ennalta asetettu tavoitevirhe.

Laskentakoodissa ROI:n säteen kasvattaminen on rajattu 50 %, jotta tehtävä pysyisi käypänä eikä alkaisi niin helposti värähtelemään. ROI:n sädettä pienennetään korkeintaan 33 %. Lisäksi ROI:n säteen alarajaksi on asetettu 0.5 %. Jos edellisen kierroksen säde on tämän alle, niin asetetaan ROI:n säteeksi 0.75 %. Tällä alarajalla on haettu lisää tarkkuutta optimointiin, ettei iterointi loppuisi liian aikaisin. Toinen syy siihen on se, etteivät muuttujien rajat menisi liian pieneksi, koska tällöin voi olla ettei saada tehtyä D-optimaalista suunnittelumatriisia laskentatarkkuuksien rajoissa. MATLABin rowexch- ja cordexch-funktiot varoittavat tällöin, että suunnittelumatriisi on singulaarinen.

Kaavoista huomataan, että lasketun suhteellisen virheen ollessa suuri, pienennetään ROI:n sädettä ja vastaavasti virheen ollessa pieni ROI:n sädettä kasvatetaan. Lisäksi havaitaan, että suuremmalla tavoitevirheellä kasvatetaan ROI:n sädettä helpommin. Kuvassa 16 on periaatekuva ROI:n muuttumista adaptiivisella ROI:n säteellä.



Kuva 16 *Käypä suunnitteluavaruus ja iteratiivinen optimointi.*

Liitteessä 10 havainnollistetaan asiaa lisää testiprobleeman TP12 (Hock & Schittkowski 1981) kautta. Tehtävää on optimoitu kvadraattisella mallilla ja Scip-algorimilla. Scip piirtää kahden muuttujan tapauksessa tasa-arvopinnat optimoinnista, jotka laskentatyökalu tallentaa tehtävän output-kansioon. Liitteen kuvissa valkoinen alue on tarkasteltava suunnitteluavaruus, pisteviiva kuvaa kohdefunktion arvoa ja katkoviiva rajoitefunktioiden muodostamia rajoja. Punainen piste kuvissa on optimointialgoritmin iteraatiokierroksella saama optimipiste.

Rajojen määrittely eroaa eri muuttujatyypeillä. Koska työkalulle voidaan antaa sekalukutehtävä, niin työkalu määrittelee rajat muuttujatyypeittäin. Kokonaislukumuuttujilla rajat pyöristetään lähimpään kokonaislukuun siten, että ROI:n koko on vähintään +/- 1 edellisen optimipisteen ympärillä. Diskreettien muuttujien kanssa toimitaan niin, että kasvatetaan iteraativisesti ROI:n sädettä kyseisen muuttujan kohdalla, kunnes vähintään kaksi arvoa osuu ROI:n sisään. Lopulta on saatu uudet rajat kyseiselle muuttujalle. Laskentatyökaluun on lisätty mallin valintaan myös vaihtoehto ”Lin/quad”, joka on tarkoitettu tällä hetkellä vain diskreettien joukkojen kanssa käytettäväksi. Tällä valinnalla lähdetään liikkeelle lineaarisella mallilla ja aina tarvittaessa iteraatiokierroksilla vaihdetaan malli kvadraattiseksi jos ROI:n sädettä diskreettien muuttujien osalta on jouduttu nostamaan diskreettien arvojen saamiseksi rajojen sisään. Tällä on haettu tehokkuutta ja tarkkuutta tehtävien ajoon.

Rajojen laskennan jälkeen määritetään koepisteet, joissa lasketaan ANSYS-analyysit. Koepisteet lasketaan D-optimaalisella koesuunnittelumenetelmällä MATLA-Bin rowexch- tai cordexch-funktiolla. Rowexch- ja cordexh-funktiot palauttavat suunnittelumatriisin ja koepistematriisin. Koesuunnittelusta ja käytetyistä funktioista on kerrottu tarkemmin luvussa 2.2.

Vaiheessa 2 ajetaan analyysit määritetyissä koepisteissä. Techila-laskentaympäristöä käytettäessä koepisteet lähetetään vapaille laskentakoneille laskettavaksi.

Vaiheessa 3 luodaan vastepinta eli sovitetaan suunnittelumatriisi havaintoihin pienimmän neliösumman menetelmällä. Vastepintamenetelmästä on kerrottu tarkemmin luvussa 2.1.

Vaiheessa 4 suoritetaan optimointi valitulla algoritmilla. Diskreettiä optimointia varten laadittiin luvussa 3.4 esitetyt diskreetit mallit, jotka implementoitiin laskentatyökaluun optimointialgoritmeja varten. Laskentatyökaluun implementoiduista optimointialgoritmeista kerrotaan luvussa 4.2. Optimointitehtävän muodostaminen vastepintamenetelmällä saadusta mallista kerrotaan luvussa 2.1. Mikäli optimointialgoritmi ei löydä käypää ratkaisua, niin lineaarisella mallilla optimoidaan heti uudestaan Interior point -algoritmilla. Kvadraattisilla malleilla muutetaan malli yhden iteraation ajaksi lineaariseksi ja valitaan algoritmiksi Interior point, jonka jälkeen palataan vaiheeseen 1. Seuraavalla iteraatiolla palataan alkuperäiseen malliin ja algoritmiin.

Vaiheessa 5 tarkastetaan, että toteutuuko jokin lopetusehto. Lopetusehtoina käytetään iteraatioiden maksimimäärää ja suhteellista konvergenssitoleranssia. Iterointi lopetetaan, kun kohdefunktio on alle asetetun konvergenssitoleranssin

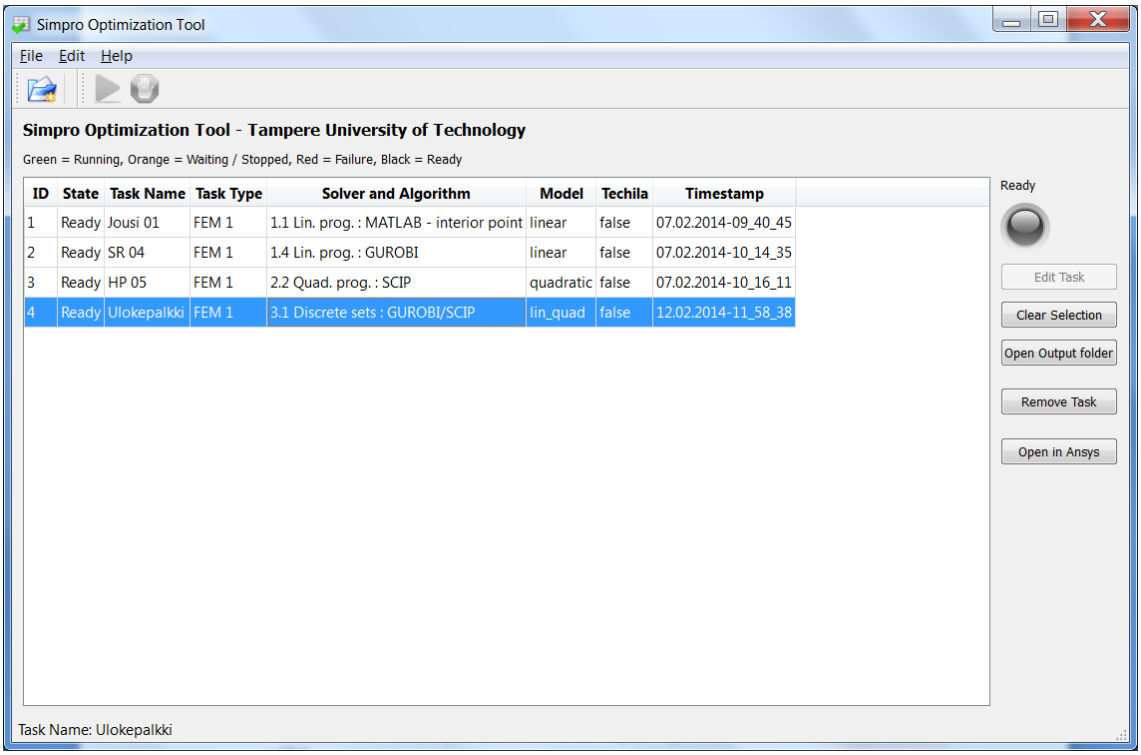
$$\frac{\hat{f}(\mathbf{x}_k^*) - \hat{f}(\mathbf{x}_{k-1}^*)}{\hat{f}(\mathbf{x}_k^*)} \leq \text{convtol}, \tag{39}$$

missä convtol on ennalta asetettu konvergenssitoleranssi,
$\mathbf{x}^* \in \mathbb{R}^n$,
k on iteraatio ja
$\hat{f}$ on mallin antama kohdefunktion arvo.

Mikäli lopetusehto ei toteudu niin luodaan väliraportteja, jonka jälkeen päivitetään ROI ja jatketaan iterointia vaiheesta 1. Lopetusehdon toteutuessa optimointi lopetetaan ja tehdään tarvittavat raportit. Mikäli algoritmiksi oli valittu Interior point ja asetettu tehtävä kokonaislukutehtäväksi ei lopetetakaan, vaan haetaan binäärioptimoinnilla jatkuvan optimipisteen ympäriltä parasta kokonaislukuratkaisua. Käytetystä binäärioptimoinnista kerrotaan artikkelissa (Pajunen & Heinonen 2013) tarkemmin. Kyseistä menetelmää testattiin muutaman tehtävän kanssa ja Tulokset-luvussa nähdään sillä saatuja tuloksia.

Jokaisen iteraation lopuksi päivitetään historiadataa ja näytetään kuvaajia. Lisäksi tallennetaan kuvaajat ja kirjoitetaan historiadataa tekstitiedostoihin. Raportoinnista on kerrottu lisää seuraavassa luvussa 4.1.3.

Optimointitehtävän lopuksi sovellus siirtää raportit tehtävän output-kansioon ja merkitsee tehtävän tilaksi ”Ready”, jonka jälkeen aloitetaan seuraavaan tehtävän suorittaminen. Mikäli kaikki tehtävät ovat valmiina, päivitetään käyttöliittymän tilat ja käyttöliittymän merkkivalo kertoo, kun kaikki tehtävät ovat valmiina. Kuvasta 17 nähdään valmistunut tehtäväjono.



Kuva 17 *Valmistunut tehtäväjono.*

Kuvasta huomataan myös, että tehtävän muokkaamiseen tarkoitettu painike on pois käytöstä. Valmistuneen tehtävän optimipisteen voi nyt viedä ANSYS-ohjelmaan analysoitavaksi valitsemalla tehtävän ja napsauttamalla painiketta ”Open in ANSYS”.

4.1.3 Tulokset ja raportointi

Työkalu kerää paljon historiadataa iterointien aikana, joiden avulla piirretään kuvaajia ja tallennetaan erilaisia raportteja tekstitiedostoihin. Iterointikierroksen jälkeen näytetään ja tallennetaan kohdefunktioiden kuvaajat sekä käyttäjän valitsemat muut vasteet. Kuvaajat piirretään mallin antamista ja todellisista vasteista. Lisäksi iterointien aikana näytetään ja tallennetaan käyttäjän valitsemat suunnittelumuuttujien kuvaajat. Lopuksi tehdään lisää raportteja ja siirretään ne tehtävän output-kansioon. Liitteessä 2 on kerrottu lyhyesti laskentatyökalun tuottamista raporteista. Tulokset-luvussa nähdään esimerkkejä tuotetuista kuvaajista.

Työkaluun on lisätty käyttäjän asettama absoluuttinen virhetoleranssi TolCon tehtävän rajoitteina toimiville vasteille. Syynä siihen on se, että approksimoidun mallin antama optimipiste ei välttämättä ole todelliselle optimointikohteelle, esimerkiksi ANSYS- mallille hyvin käypä. Näin voi käydä jos approksimoitu malli ei kuvaa hyvin todellisia vasteita viritetyssä aliavaruudessa. Syynä tähän voi olla liian suuri tarkasteltava aliavaruus ja liian pieni approksimoinnin aste. Lisäksi iterointikierroksien aikana on voinut löytyä parempi lokaali optimipiste kuin mihin iterointi on päättynyt. Näistä syistä johtuen ei voida vain lajitella lokaaleja tuloksia paremmuusjärjestykseen, koska paras tulos voi olla optimointikohteelle erittäin huonosti käypä. Tästä syystä työkaluun annetaan edellä mainittu virhetoleranssi, jonka avulla lajitellaan paremmuusjärjestykseen virhetoleranssin alle jääneet iterointikierrosten aikana löytyneet lokaalit ratkaisut. Työkalun antama paras ratkaisu on siten, parhaimman tuloksen antama piste, joka ei riko asetettua virhetoleranssia. Työkalun käyttäjä voi katsoa myös tarkemmin raportoituja tuloksia ja valita niistä tarpeisiin riittävän tarkan ratkaisun.

Jokaisen iteraatiokierroksen lopuksi lasketaan siten keskimääräinen rajoitusyhtälöiden ylitys $\overline{\boldsymbol{gviol}}$ (40) mallin antamille ja todellisille vasteille, jonka avulla merkitään saatu lokaali optimi käyväksi tai epäkäyväksi. Rajoitusyhtälöiden rikkomisen kriteerinä voitaisiin käyttää myös suurinta rajoitusyhtälöiden ylitystä, mutta tähän valittiin keskimääräinen ylitys. Rajoitusyhtälöiden ylitys ja keskimääräinen ylitys on

$$\begin{gathered} \boldsymbol{gviol} = |\mathbf{b} - \mathbf{g}| \\ \overline{\boldsymbol{gviol}} = \frac{1}{\mathrm{m}} \sum_{i=1}^{m} gviol_i \\ gviol_i = 0, \text{ kun } g_i \leq b_i \\ i = 1 \ldots m, \end{gathered} \tag{40}$$

missä **b** = rajoitusyhtälöiden oikea puoli
g = rajoitefunktioiden vasteet
m = rajoitusyhtälöiden lukumäärä

Iteraatiokierroksella saatu optimipiste merkitään käyväksi, kun

$$\overline{\boldsymbol{gviol}} \leq \text{TolCon}, \tag{41}$$

missä TolCon on käyttäjän asettama absoluuttinen virhetoleranssi. Approksimoidun mallin pitäisi pysyä menetelmästä johtuen aina käypänä. Mallin vasteet pysyvät yleensä käypänä, koska optimia haetaan aina käyvän pisteen lähiympäristöstä. Todelliset vasteet taas eivät välttämättä ole käypiä jos malli ei vastaa hyvin todellista ilmiötä. Mallissa rikotaan tätä virhetoleranssia vain kahdessa tapauksessa. Luvussa 3.2 kerrottiin, että käypä joukko voi olla tyhjä. Näin käy yleensä jos lähdetään epäkäyvästä pisteestä, mutta näin voi tapahtua myös iterointien aikana jos malli approksimoi huonosti todellisia vas-

teita. Käyvän joukon ollessa tyhjä, niin käytettävistä olevista menetelmistä kokemusten mukaan ainoastaan MATLABin Interior point -algoritmi palauttaa parhaiten käyvän ratkaisun muuttujien rajojen sisältä. Tämä ratkaisu rikkoo kuitenkin rajoitusyhtälöitä, jolloin se merkataan mallin kuvaajiin epäkäyväksi. Ainoastaan jos asetettu virhetoleranssi on erittäin suuri, niin tämä piste voidaan merkitä käyväksi. Toinen tapaus missä mallin antama ratkaisu merkataan iterointien aikana epäkäyväksi, on se, että jos toleranssi on pieni verrattuna optimointialgoritmien omiin virhetoleransseihin. Optimointialgoritmeissa on tietyt oletusarvot esimerkiksi niiden omalle TolCon-toleranssille, joita voidaan tarvittaessa muuttaa. Käytännössä työkaluun asetettua virhetoleranssia rikotaan yleensä vain todellisten vasteiden kanssa, ellei sitten lähdetä epäkäyvästä pisteestä tai olla asetettu liian pieni virhetoleranssi laskentatyökaluun.

Vasteen arvo merkitään vasteiden kuvaajiin punaisella täplällä jos saatu optimipiste on epäkäypä virhetoleranssiin nähden. Virhetoleranssi ei siis vaikuta mitenkään optimointiin eli sen ylitys ei keskeytä iterointia. Kuvaajien avulla käyttäjä voi halutessaan keskeyttää iteroinnin, esimerkiksi silloin jos todelliset vasteet rikkovat käyttäjän asettamaa virhetoleranssia useamman kierroksen aikana peräkkäin. Tämän jälkeen käyttäjä voi esimerkiksi vaihtaa kvadraattiseen malliin tai pienentää ROI:n alkusädettä ja ajaa tehtävän uudestaan.

Täysin käypänä pidetään tässä työssä ratkaisua, jossa ei rikota keskimääräisesti yhtään rajoitusyhtälöitä. Tässä tulevat tietenkin laskentatarkkuudet ottaa huomioon MATLAB- ja ANSYS-ohjelman osalta. Oletusasetuksilla ANSYS palauttaa työssä tehtyjen havaintojen mukaan vasteet 1e-16 tarkkuudella, jotka viedään sitten MATLABiin. MATLABin tuottamaan raporttiin tulee keskimääräinen rajoitusyhtälöiden ylitys 1e-12 tarkkuudella, joka on käytännön tehtäviä varten todennäköisesti turhankin tarkka. Vasteiden arvot tallennetaan talteen myös tässä samassa tarkkuudessa. Tässä työssä pidetään yleisesti ratkaisua täysin käypänä jos rikotaan rajoitusyhtälöitä keskimäärin alle 1e-12. ANSYS-ohjelmalle koepisteet puolestaan lähetetään annetun suunnittelumuuttujien tarkkuuden mukaan, koska käytännössä on havaittu, että tietyissä tehtävissä, jotkut muuttujat eivät saa olla kuin esimerkiksi 1e-4 tarkkuudella. Edellä mainitut tarkkuudet ovat siis suunnittelumuuttujiin nähden turhankin tarkkoja käytännön tehtäviin. Testiprobleemien optimoinnissa käytettiin maksimitarkkuutta, jotta päästiin mahdollisimman tarkkoihin tuloksiin.

Virhetoleranssi on tällä hetkellä siis absoluuttinen arvo, jolloin pienillä vasteiden arvoilla joutuu asettamaan pienen virhetoleranssin. Tässä mielessä on perusteltua olla mahdollisuus näinkin suureen tarkkuuteen kuin 1e-12. Laskentakoodiin ei ole tehty tällä hetkellä skaalausta, koska skaalaus tehdään usein FEM-suunnitteluohjelmassa.

4.2 Optimointialgoritmien implementointi

Tässä luvussa kerrotaan työkaluun implementoitujen optimointialgoritmien valintaperusteista ja löydetyistä muista tarkemman tutustumisen arvoisista vaihtoehdoista. Tässä työssä yhtenä tavoitteena oli selvittää erilaisten optimointimenetelmien sopivuutta vai-

heittaiseen vastapintamenetelmään ja optimointiin. Lisäksi tavoitteena oli selvittää 2. asteen polynomiapproksimaation tuomaa hyötyä tutkittavassa menetelmässä, jolloin tarvittiin siihen sopivaa optimointialgoritmia.

Lineaarisen metamallin optimointiin päätettiin kokeilla kaikkia MATLABin linprog-funktion sisältämiä optimointimenetelmiä. Kyseiset menetelmät ovat hyvin perinteisiä ja paljon käytettyjä lineaariseen optimointiin. MATLABin linprog-funktiolla voidaan ratkaistava vain jatkuvia tehtäviä. (Matlab 2013)

Kvadraattiseen optimointiin puolestaan ei löytynyt MATLABista sopivaa funktiota. Kvadraattiseen optimointiin tarkoitettu quadprog-funktio ei sopinut, koska siinä rajoitteiden pitää olla lineaariset. MATLABissa on epälineaarisiin rajoitteellisiin tehtäviin tarkoitettu fmincon-funktio, mutta koska sitä ei varsinaisesti ole tarkoitettu kvadraattisiin QCQP-tehtäviin, päätettiin etsiä muita vaihtoehtoja. (Matlab 2013) Koska käyttöliittymän ohjelmointi vei työn alkuaikoina suurimman osan ajasta, täytyi haetun kvadraattisen algoritmin olla helposti implementoitavissa MATLABin laskentakoodiin.

Akateemiseen käyttöön ilmainen ja MATLABin kautta helposti käytettävä Gurobi vaikutti hyvin lupaavalta, koska sillä voidaan laskea myös sekalukutehtäviä ja sen luvattiin ratkaisevan tehokkaasti kvadraattisia tehtäviä (Gurobi 2014). Gurobin avulla saatiin implementoitua kvadraattinen metamalli laskentatyökaluun ja kokeiltua kvadraattisen mallin optimointia. Ensimmäiset kokeilut menivät sillä hyvin, mutta heti hieman vaativammassa tehtävässä Gurobi kaatui kvadraattisten funktioiden negatiiviseen definiittisyyteen. Selvisi, että Gurobi on tarkoitettu vain konvekseihin QCQP-tehtäviin. Gurobin lineaarinen algoritmi puolestaan toimi hyvin ja ennen kaikkea sen pystyessä ratkaisemaan sekalukutehtäviä, päätettiin jättää se työkaluun.

Lopulta löytyi akateemiseen käyttöön ilmainen Scip-algoritmi, joka tulee OPTI Toolboxin (OPTI 2014) mukana, joka on laaja kokoelma optimointialgoritmeja MATLAB-ohjelmasta ajettaviksi. Scip-algoritmi käyttää hajota ja rajoita -menetelmää ja sen luvataan ratkaisevan myös epäkonvekseja QCQP-tehtäviä löytäen niissä myös globaalin ratkaisun. Lisäksi Scip-algoritmi pystyy ratkaisemaan sekalukutehtäviä, joten löytyi tarpeet täyttävä algoritmi, jonka tehokkuus todettiin heti hyväksi. Mainittakoon, että sekä Scip että Gurobi ovat kaupalliseen käyttöön hyvin kalliita, mutta koska tavoitteena oli kokeilla monipuolisesti erityyppisiä optimointitehtäviä vastepintamenetelmällä, niin päätettiin kokeilla näillä aluksi. Mikäli aikaa olisi ollut enemmän käytössä optimointialgoritmien etsimiseen ja sen implementoimiseen niin olisi ehdottomasti kannattanut kokeilla ilmaisia vapaan lähdekoodin ratkaisuja lisäksi. Testiprobleemia ja ANSYS-tehtäviä varten, joiden tuloksia seuraavassa luvussa esitetään, saatiin kuitenkin hyvä algoritmivalikoima testattavaksi.

Taulukosta 4 nähdään käytettyjä lyhenteitä, joista osaa tässä työssä jatkossa käytetään. Lisäksi taulukosta nähdään työkaluun lisätyt algoritmit niiden alkuperäisellä nimellä.

Taulukko 4 *Käytetyt lyhenteet ja käytetyt optimointialgoritmit.*

Lyhenne	Kuvaus	
M	Malli	
L tai Lin.	Mallin tyyppi	Lineaarinen malli
Q	Mallin tyyppi	Kvadraattinen malli
PQ	Mallin tyyppi	Puhtaasti kvadraattinen malli
Iter	Iteraatio	Paras iteraatio / iteraatioiden lukumäärä
C	Muuttujan tyyppi	Jatkuva muuttuja
I	Muuttujan tyyppi	Kokonaislukumuuttuja
B	Muuttujan tyyppi	Binäärimuuttuja
S	Muuttujan tyyppi	Riippumaton diskreetti muuttuja
D	Muuttujan tyyppi	Riippuva diskreetti muuttuja
ML IP	Optimointialgoritmi	MATLAB: Interior point, large-scale, default settings
ML IP + DH	Optimointialgoritmi	Jatkuvan optimin jälkeen diskreetin pisteen haku
ML SX	Optimointialgoritmi	MATLAB: Simplex, medium-scale, default settings
ML AS	Optimointialgoritmi	MATLAB: Active-set, medium-scale, default settings
Gurobi	Optimointialgoritmi	Gurobi: Automatic algorithm
Scip	Optimointialgoritmi	SCIP: Spatial Branch and Bound using IPOPT & SoPlex
ML FMM	Optimointialgoritmi	MATLAB: fminimax
DS G/S	Optimointialgoritmi	Gurobi/SCIP (diskreeteille joukoille)
DS FMM	Optimointialgoritmi	MATLAB: fminimax (diskreeteille joukoille)
A	Rajoite	Epäyhtälörajoite
Aeq	Rajoite	Yhtälörajoite
Qc	Rajoite	Kvadraattinen epäyhtälörajoite
Epäkonv QP		Voidaan käyttää epäkonvekseissa kvadraattisissa tehtävissä.
MO-menetelmä		Multi-objective optimization eli monitavoiteoptimoinnin menetelmä
SO / MO		Single objective / Multiobjective Yksi kohdefunktio / Monitavoitetehtävä
Lin. skal.	MO-menetelmä	Linear scalarization
Pre-empt.	MO-menetelmä	Pre-emptive
Minimax	MO-menetelmä	

Laskentatyökalun ohjeisiin eli Help-sivuille on lisätty sopivan algoritmin valitsemista varten ohjetaulukko. Lisäksi siinä kerrotaan, mitä rajoitusyhtälöitä eri algoritmeille voidaan antaa. Vastaava ohje on taulukossa 5. Sitä voidaan lukea käyttämällä taulukkoa 4 hyödyksi.

Taulukko 5 *Laskentatyökaluun implementoidut algoritmit ja niiden käyttö.*

M	Algoritmi / muuttuja-tyypit	Raj.	LP	MILP	QC-QP	MI-QCQP	Epä-konv QP	SO / MO	MO-menetel-mä
L	ML IP / C	A ja/tai Aeq	x	-	-	-	-	x / x	Lin. scal. Pre-empt.
L	ML SX / C	A ja/tai Aeq	x	-	-	-	-	x / x	Lin. skal.
L	ML AS / C	A ja/tai Aeq	x	-	-	-	-	x / x	Lin. skal.
L PQ Q	Gurobi / CIB	(A ja/tai Aeq) tai Qc	x	x	x	x	-	x / x	Lin. skal. Pre-empt.
PQ Q	Scip / CIB	Qc	-	-	x	x	x	x / x	Pre-empt.
L PQ Q	ML FMM / C	(A ja/tai Aeq) tai Qc	x	-	x	-	x	- / x	Minimax
L PQ Q	DS G/S / CIBSD	(A ja/tai Aeq) tai Qc	x	x	x	x	x	x / x	Lin. skal. Pre-empt.
L PQ Q	DS FMM / CSD	A ja/tai Aeq) tai Qc	x	-	x	x	x	- / x	Minimax

Implementoituihin optimointialgoritmeihin liittyy myös paljon asetuksia, joita ei vielä nykyisellä versiolla voida muuttaa käyttöliittymän kautta. Optimointialgoritmeissa on käytetty tällä hetkellä pääasiassa niihin liittyviä oletusasetuksia. Algoritmien asetuksia muuttamalla voidaan vaikuttaa esimerkiksi lopetusehtoihin ja käypyystoleransseihin. Oletusasetuksilla päästään kuitenkin yleensä hyviin tuloksiin. Ainoastaan Gurobin kvadraattiseen optimointiin on muutettu hieman oletusasetuksia, jotta optimointi ei kaatuisi niin helposti kvadraattisten matriisien negatiiviseen definiittisyyteen.

Gurobi on tarkoitettu konveksien kvadraattisten tehtävien optimointiin, joten laskentatyökalulla sitä ei kannata käyttää kvadraattisen mallin optimointiin, koska iteraatioiden aikana mallista voi tulla epäkonveksi vaikka tehtävä näyttäisi aluksi menevän hyvin. Kvadraattisiin tehtäviin kannattaa työkalusta valita Scip-algoritmi, joka osaa kuten edellä mainittiin, ratkaista myös epäkonveksin tehtävän. Lisäksi sen luvataan löytävän globaalin optimin kvadraattisesta tehtävästä.

Tämän luvun lopuksi käsitellään vielä muutamaa algoritmia, joita työkaluun voidaan halutessa helposti lisätä. Yksi on jo edellä mainittu MATLABin fmincon-funktio. Yhtenä käytännön toteutuserona verrattuna käytettyihin muihin algoritmeihin on se, että fmincon-funktiolle ei voida antaa parametrina suoraan lineaarisia termejä vektoreina ja kvadraattisia matriiseina. Kohdefunktio ja epälineaariset rajoitefunktiot annetaan fmincon-funktiolle MATLABin funktiona. Tämä ei kuitenkaan käytännössä tuota ongelmia.

Vastaavasti annetaan laskentatyökaluun jo implementoidulle fminimax-funktiolle approksimoidut mallit. Funktion antamisessa algoritmille on hyvänä puolena se, että funktiossa voi olla esimerkiksi metamallinnuksella muodostettu neuroverkko, joka palauttaa algoritmille kysytyt vasteet. Fmincon-funktioon on valittavissa useampi algoritmi, joista Interior point voisi olla tehokas, mikäli se osaa hakea käypää aluetta kuten lineaarisen algoritmin vastaava. Käytännön tehtäviä varten Fmincon-funktiossa on se huono puoli, että se on tarkoitettu ainoastaan jatkuviin tehtäviin. Kokonaislukuoptimi voidaan hakea jatkuvasta optimista myös erikseen, esimerkiksi binäärioptimoinnilla, jota on käytetty lineaarisen Interior point -algoritmin kanssa. Sekalukutehtäviin tämä ei kuitenkaan hyvin sovi. Fmincon kannattaa implementoida työkaluun, koska se tulee MATLABin mukana. Tällöin voidaan tehdä kvadraattista optimointia ilman Scip-algoritmia.

MATLABista löytyy myös geneettiseen optimointiin tarkoittu ga-funktio, johon annetaan optimointiparametrit vastaavasti kuin fmincon-funktioon. Ga-funktioon liittyy kuitenkin paljon itse menetelmään liittyviä asetuksia, jotka eivät välttämättä ole oletuksina aina tehtävään sopivia. Ga-funktiolle voidaan määritellä muuttujakohtaisesti, että onko se jatkuva vai kokonaisluku. Eli se sopisi myös sekalukutehtäviin. Tämäkin funktio on hyvä lisätä työkaluun vaihtoehdoksi. Ga-funktio sopii erityisesti erittäin epälineaarisiin tehtäviin. (Matlab 2013)

Avoimen lähdekoodin algoritmeista kokeilemisen arvoiselta vaikuttaa ainakin ipopt (Interior Point OPTimizer, pronounced eye-pea-Opt), joka on tarkoitettu epälineaarisiin rajoitteita sisältäviin tehtäviin eikä se vaadi konveksisuutta. Scip-algoritmi käyttää ipot-algoritmia hyödyksi omassa ratkaisussaan. Hyvänä puolena siinä on sekin, että siinä on valmis rajapinta MATLABiin ja sen saa edellä mainitun OPTI Toolboxin mukana. (OPTI 2014)

5 TULOKSET

Seuraavaksi esitellään työkalulla saatuja tuloksia. Ensin kerrotaan, että miten tuloksia on tässä tulkittu. Tuloksien esittämisen ja analysoinnin jälkeen on vielä yhteenveto tuloksista. Yhteenvedossa esitetään myös kehitysideoita.

5.1 Tuloksien tulkinta

Tuloksissa on annettu paras käypä ratkaisu annetun virhetoleranssin TolCon sisällä ja lisäksi pienimmän arvon antanut epäkäypä ratkaisu, jos se on pienempi kuin paras käypä ratkaisu. Jos käypää ratkaisua ei ole löytynyt, annetaan parhaimpana ratkaisuna piste, jossa kohdefunktion arvo on pienin. Laskentatyökalu tallentaa tuloksia tekstitiedostoina. Käypyys ja rajoitusyhtälöiden rikkominen on merkitty ”top_list.txt”-tiedostossa iteraation kohdalle omiin sarakkeisiin. MATLABin tuottamaan raporttiin tulee keskimääräinen ylitys 1e-12 tarkkuudella. Tässä työssä pidetään ratkaisua täysin käypänä jos rikotaan rajoitusyhtälöitä keskimäärin alle 1e-12. Todellisissa tehtävissä oletettavasti ei tarvita läheskään näin tarkkoja tuloksia. Suunnittelija voi valita tuloksista käypyystarpeen mukaan parhaimman tuloksen. Alun perin on voitu asettaa liian tiukka käypyysvaatimus, mutta siitä voidaan ajamisen jälkeen vielä tinkiä. Itse optimointiin käypyysvaatimus ei vaikuta, sillä algoritmeissa on omat käypyysvaatimukset kuten luvussa 4.1.3 kerrottiin. Lisäksi ”output_history.txt”-raporttiin tallentuu kaikkien vasteiden arvot iteraatiokierroksittain. Siitä voidaan tarkastella yksittäisten rajoitteiden ylittymistä ja valita tarvittaessa siten parhaiten sopiva ratkaisu. Keskimääräinen ylitys ei välttämättä aina ole hyvä tapa tarkastella ratkaisua. Algoritmeissa usein käytetty maksimiylitys voisi olla parempi käypyystoleranssi.

Testiprobleemien tuloksien vertaileminen on tarkkuutta haettaessa usein hankalaa, jos ei tiedetä vertailtavalle kirjallisuudesta saadulle tulokselle rajoitusyhtälöiden rikkomista. Suurimmassa osassa testiprobleemissa sitä ei ollut kerrottu. Tällöin ei voida varmasti sanoa, että päästiinkö parempaan tulokseen kuin on kirjallisuudessa päästy, koska sallimalla suurempi rajoitusyhtälöiden ylitys, päästään yleensä parempaan tulokseen.

Tehtäviin kulunut aika on vain suuntaa antava, koska suoritusaika riippuu myös tietokoneen muusta käytöstä ja Techila-laskentaympäristöä käytettäessä vapaista laskentayksiköistä. Täten aikaa ei voida suoraan verrata toisiinsa. Iteraatioiden lukumäärä kertoo siten paremmin metamallin ja optimointialgoritmin sopivuudesta tehtävään. Esitetyissä testiprobleemissa on piirretty kaikista funktioista kuvaajat. Kuvien piirtäminen hidastuttaa hieman iterointia, varsinkin jos kuvia on paljon. Ilman kuvien piirtoa testiprobleemat valmistuisivat hetkessä. Käytännön tehtävissä kuvien piirtämiseen menneellä ajalla ei ole merkitystä, sillä vasteiden laskenta on niissä aikaa vievin.

Ensimmäiset mallin vasteet on laskettu alkuarvoilla ensimmäisen iteraation mallin avulla. Tästä syystä aloituspiste ei välttämättä ole mallille käypä vaikka se olisi todellisille vasteille käypä. Seuraavat vasteet on laskettu samalla kierroksella saadun mallin ja optimipisteen avulla. Tuloksissa ensimmäinen piste on aloituspiste, jota sitten on iteratiivisesti optimoitu.

5.2 Laskentatyökalun validointi testiprobleemien avulla

Testiprobleemien tavoitteena oli selvittää, että sopiiko laskentatyökalu ja siinä käytetty vastepintamenetelmä simulaatiopohjaiseen optimointiin. Vaatimuksena sille on, että työkalulla pitää pystyä optimoimaan hyvin erityyppisiä optimointitehtäviä mahdollisimman tarkasti. Luvussa 3 todettiin, että käytännön tehtävät ovat usein epälineaarisia sekalukutehtäviä. Kirjallisuudesta etsittiin siten erityyppisiä testiprobleemia, joita voi myös käytännössä tulla vastaan.

Seuraavaksi päätettiin, että millä työkalun parametreilla testiprobleemia optimoidaan ja mitä arvoja niihin kiinnitetään. Tunnistettiin, että kehitetyssä laskentaproseduurissa vastepintamenetelmällä saadun mallin tarkkuuteen vaikuttaa sen approksimaation aste sekä aliavaruuden koko. Aliavaruuden kokoon puolestaan vaikuttaa menetelmässä käytettävä tavoitevirhe. Mallin optimointiin vaikuttaa taas käytettävä optimointimenetelmä. Huomataan, että testattavia vaihtoehtoja on paljon ja kaikkia mahdollisia kombinaatioita ei voida kokeilla. Kokemukseen perustuen on kuitenkin todettu usein hyvin toimivaksi 10 % ROI:n alkusäde ja 1 % tavoitevirhe. Taulukosta 6 nähdään valitut testiparametrit, joita käytettiin testiprobleemien optimoinnissa. Tärkein selvitettävä asia on se, että kuinka hyviin tuloksiin päästään eri mallien kanssa.

Taulukko 6 *Testiprobleemien optimoinnissa käytetyt laskentatyökalun parametrit.*

Testattava parametri	Vaihtoehdot
Aloitus ROI:n säde	10, 20, 50 %
Tavoitevirhe	1, 5, 10 %
Muuttujan tyyppi	Jatkuva, kokonaisluku, diskreetti (riippumaton ja riippuva)
Malli	Lineaarinen (L) Kvadraattinen (Q) Puhtaasti kvadraattinen (PQ)
Algoritmi	MATLAB Interior point (L) MATLAB Simplex (L) MATLAB Active-set (L) Gurobi Automatic algorithm (L) SCIP (Q, PQ)

Laskentatyökalua on testattu sen kehittämisen aikana kymmenillä erilaisilla testiprobleemoilla. Työssä käytettäväksi valittiin niistä 10 erityyppistä tehtävää. Taulukossa 7 on

esitelty testiprobleemien kuvauksissa käytetyt lyhenteet ja taulukossa 8 esitellään tarkasteltavat testiprobleemat.

Taulukko 7 *Testiprobleemien kuvauksissa käytetyt lyhenteet.*

Lyhenne	Kuvaus
n	Muuttujien lukumäärä
f	Kohdefunktio: q on kvadraattinen nl on epälineaarinen
ml	Lineaaristen rajoitusyhtälöiden lukumäärä
mq	Kvadraattisten rajoitusyhtälöiden lukumäärä
mnl	Epälineaaristen rajoitusyhtälöiden lukumäärä

Taulukko 8 *Testiprobleemat.*

Kohde (tunniste)	Kuvaus	n	Muuttujan tyyppi	f	ml	mq	mnl
Kierrejousi (Jousi)	Hyvin epälineaariset kohde- ja rajoitefunktiot	3	C	nl	1	-	3
Hitsattu palkki (HP)	Hyvin epälineaariset kohde- ja rajoitefunktiot	4	C	nl	1	-	4
Nopeudenalennin (SR)	Hyvin epälineaariset kohde- ja rajoitefunktiot	7	C	nl	-	5	6
TP12	Kokonaislukutehtävä	2	I	q	-	1	-
E3	Kokonaislukutehtävä	2	I	q	1	1	-
Painesäiliö (PV)	Riippumattomia diskreettejä joukkoja	4	S	nl	3	-	1
Ulokepalkki 1 (CB_01)	Sekalukutehtävä jossa diskreettejä joukkoja	10	IISSSSCCCC	q	5	-	6
Ulokepalkki 2 (CB_02)	Sekalukutehtävä jossa riippuvia diskreettejä joukkoja	10	IIDDDDCCCC	q	5	-	6
Ulokepalkki 3 (CB_03)	Sekalukutehtävä jossa riippuvia ja riippumattomia diskreettejä joukkoja	10	IISDDSCCCC	q	5	-	6
TP113	QCQP	10	C	q	3	5	-

Ulokepalkkitehtävissä on kokeiltu erityyppisiä suunnittelumuuttujia. Nämä perustuvat kuitenkin samaan optimointiprobleemaan.

Seuraavaksi esitellään saadut tulokset. Ensimmäistä testiprobleemaa analysoidaan tarkemmin. Tilan säästämiseksi kuvaajat on pienennetty yleensä aika pieniksi, mutta niistä nähdään kuitenkin suppeneminen. Ensimmäisestä testiprobleemasta näytetään pari kuvaa isompana esimerkin vuoksi. Ensimmäisellä tehtävällä on tutkittu tarkemmin edellä mainittuja testiparametreja, sillä on ajettu yhteensä 28 erilaisella asetuksella olevaa tehtävää. Vastaavilla asetuksilla on ajettu myös ensimmäinen ANSYS-mallin optimointi. Muista testiprobleemeista analysoidaan lähinnä vain mallin ja algoritmin vaikutusta

tehokkuuteen ja tarkkuuteen. Testiprobleemat on ratkaistu myös suoraan Scip-algoritmilla, jotta voidaan tarkastella sen tarkkuutta.

5.2.1 Kierrejousi

Ensimmäisessä testiprobleemassa minimoidaan kierrejousen tilavuutta, jota kuormitetaan vakiokuormalla. Kohdefunktio ja rajoitteet ovat (Lemonge et al. 2010)

$$
\begin{aligned}
&V(\mathbf{x}) = (x_1 + 2)x_2x_3^2 \\
&g_1(\mathbf{x}) = 1 - \frac{x_2^3x_1}{71785x_3^4} \leq 0 \\
&g_2(\mathbf{x}) = \frac{4x_2^2 - x_3x_2}{12566(x_2x_3^3 - x_3^4)} + \frac{1}{5108x_3^2} - 1 \leq 0 \\
&g_3(\mathbf{x}) = 1 - \frac{140.45x_3}{x_2^2x_1} \leq 0 \\
&g_4(\mathbf{x}) = \frac{x_2+x_3}{1.5} - 1 \leq 0 \\
&2 \leq x_1 \leq 15,\ 0.25 \leq x_2 \leq 1.3,\ 0.05 \leq x_3 \leq 2.
\end{aligned}
\tag{42}
$$

Kaikki muuttujat pidetään tässä jatkuvina. Kierrejousen optimointitehtäviin asetettiin tiukka lopetusehto, koska haluttiin päästä mahdollisimman tarkkoihin tuloksiin. Iteraatioiden maksimimäärä rajoitettiin 50. Käytetyt asetukset ja lähtötiedot nähdään taulukoista 9-10. Taulukosta 10 nähdään myös ajojen tulokset. Käypänä pidetään ratkaisua, jonka rajoitusyhtälöiden keskimääräinen virhe on alle asetetun toleranssin. Tuloksia on analysoitu lyhyesti liitteessä 11. Liitteissä 12-14 ovat jokaisen ajon vasteiden ja suunnittelumuuttujien kuvaajat.

Taulukko 9 *Kierrejousen optimoinnissa käytetyt yleiset asetukset ja lähtötiedot.*

Faktoreiden lukumäärä ja tyypit	3 jatkuvaa muuttujaa
Rajoitusyhtälöiden lukumäärä	4
Minimoitava vaste $V(\mathbf{x})$	Tilavuus
Aloituspiste $\mathbf{x}_0$	[10; 0.9; 0.1]
Aloituspiste käypä	kyllä
Arvot aloituspisteessä	$V(\mathbf{x}_0)$ = 0.108 $g_1(\mathbf{x}_0)$ = -0.015532 $g_2(\mathbf{x}_0)$ = -0.667077 $g_3(\mathbf{x}_0)$ = -0.733951 $g_4(\mathbf{x}_0)$ = -0.333333
Rajoitusyhtälöiden oikea puoli	**0**
Globaalit rajat	$x_1 \in [2, 15]$, $x_2 \in [0.25, 1.3]$, $x_3 \in [0.05, 2]$
Lopetusehto (konvergenssi)	1e-6 = 0.0001 %
Maksimi iteraatiot	50
Aloitus ROI (säde)	10 %

Adaptiivinen ROI	kyllä
Suunnittelumuuttujien tarkkuus	1e-12
Tavoitevirhe (TolRel)	1 %
TolCon	1e-4
Techila käytössä	ei

Taulukko 10 *Kierrejousen optimointiajot.*

Tun-niste	M	Algo-ritmi	Kuvaus	V	Käypä k/e (virhe)	Aika / s	Iter
Jousi 1	L	ML IP	Katso taulukko 9	0.012741	k (5e-5)	223	50/50
Jousi 2	L	MLSX	Katso taulukko 9	0.012792	k (5e-5)	296	50/50
Jousi 3	L	ML AS	Katso taulukko 9	0.012792	k (5e-5)	219	50/50
Jousi 4	L	Gurobi	Katso taulukko 9	0.012792	k (5e-5)	219	50/50
Jousi 5	Q	Scip	Katso taulukko 9	0.012676	k (8e-6)	215	44/45
Jousi 6	PQ	Scip	Katso taulukko 9	0.012831	k (5e-5)	227	49/50
Jousi 7	L	ML IP	Jousi 1, kiinteä ROI	0.014648	k (0)	224	6/50
Jousi 8	L	ML IP	Jousi 1, kiinteä ROI 20 %	0.019818	k (0)	233	3/50
Jousi 9	L	ML IP	Jousi 1, kiinteä ROI 50 %	0.108000	k (0)	175	1/50
Jousi 10	L	ML IP	Jousi 1, TolRel 5 %	0.012741	k (5e-5)	223	50/50
Jousi 11	L	ML IP	Jousi 1, TolRel 10 %	0.012741	k (5e-5)	223	50/50
Jousi 12	Q	Scip	Jousi 5, kiinteä ROI	0.012683	k (0)	252	17/50
Jousi 13	Q	Scip	Jousi 5, kiinteä ROI 20 %	0.013281	k (0)	251	33/50
Jousi 14	Q	Scip	Jousi 5, kiinteä ROI 50 %	0.108000	k (0)	249	1/50
Jousi 15	Q	Scip	Jousi 5, TolRel 5 %	0.012700	k (2e-6)	175	50/50
Jousi 16	Q	Scip	Jousi 5, TolRel 10 %	0.012709	k (2e-6)	177	50/50
Jousi 17	PQ	Scip	Jousi 6, kiinteä ROI	0.013184	k (0)	171	45/50
Jousi 18	PQ	Scip	Jousi 6, kiinteä ROI 20 %	0.023020	k (0)	178	3/30
Jousi 19	PQ	Scip	Jousi 6, kiinteä ROI 50 %	0.108000	k (0)	242	1/50
Jousi 20	PQ	Scip	Jousi 6, TolRel 5 %	0.012831	k (5e-5)	222	49/50
Jousi 21	PQ	Scip	Jousi 6, TolRel 10 %	0.012831	k (5e-5)	165	49/50
Jousi 22	L	ML IP	Jousi 7, aloituspiste epä-käypä $\mathbf{x}_0$ = (9, 0.8, 1)	8.800000	e (3e-1)	253	1/50
Jousi 23	Q	Scip	Jousi 12, aloituspiste epä-käypä $\mathbf{x}_0$ = (9, 0.8, 1)	8.800000	e (3e-1)	257	1/50
Jousi 24	PQ	Scip	Jousi 17, aloituspiste epä-käypä $\mathbf{x}_0$ = (9, 0.8, 1)	8.800000	e (3e-1)	255	1/50
Jousi 25	L	ML IP	Jousi 22, ROI 50 %	0.002675	e (2e-1)	238	9/50
Jousi 26	Q	Scip	Jousi 23, ROI 50 %	0.013051	k (0)	250	20/50
Jousi 27	PQ	Scip	Jousi 24, ROI 50 %	0.003769	e (2e-1)	243	15/20
Jousi 28	Q	Scip	Jousi 5. iter. max 200, convtol = 1e-12.	**0.012664**	k (2e-5)	945	118/ 200

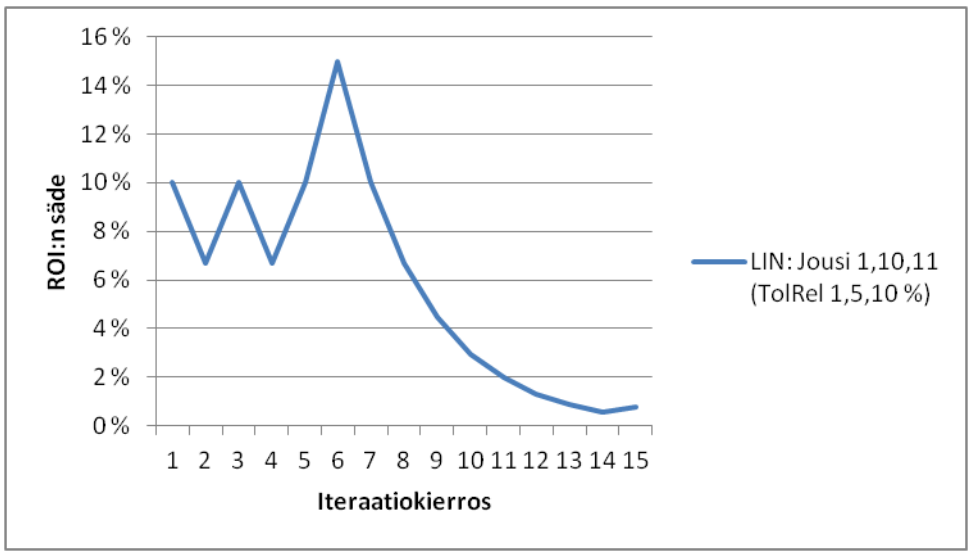
Kierrejousen optimoinnin ajolistauksesta huomataan, että rajoitusyhtälöitä rikotaan vain vähän kaikkien ajojen parhaimmassa löydetyssä pisteessä. Liitteenä olevista kuvaajista

nähdään, että taulukon 9 lähtötiedoilla ajetuissa ajoissa (Jousi 1-6) kohdefunktio suppeni kaikilla ensimmäisillä iteraatioilla hyvin. Adaptiivinen ROI:n säde näyttäisi toimivan tässä hyvin. Tiukan lopetusehdon takia iterointi jää löydetyn optimipisteen lähelle kunnes iteraatioille asetettu maksimiraja lopettaa iteroinnin. Lineaaristen mallien kanssa kohdefunktio suppeni hieman tehokkaammin. Tosin, lähellä löydettyä optimipistettä lineaarisen mallin kanssa tulee enemmän virhettä kuin kvadraattisella mallilla.

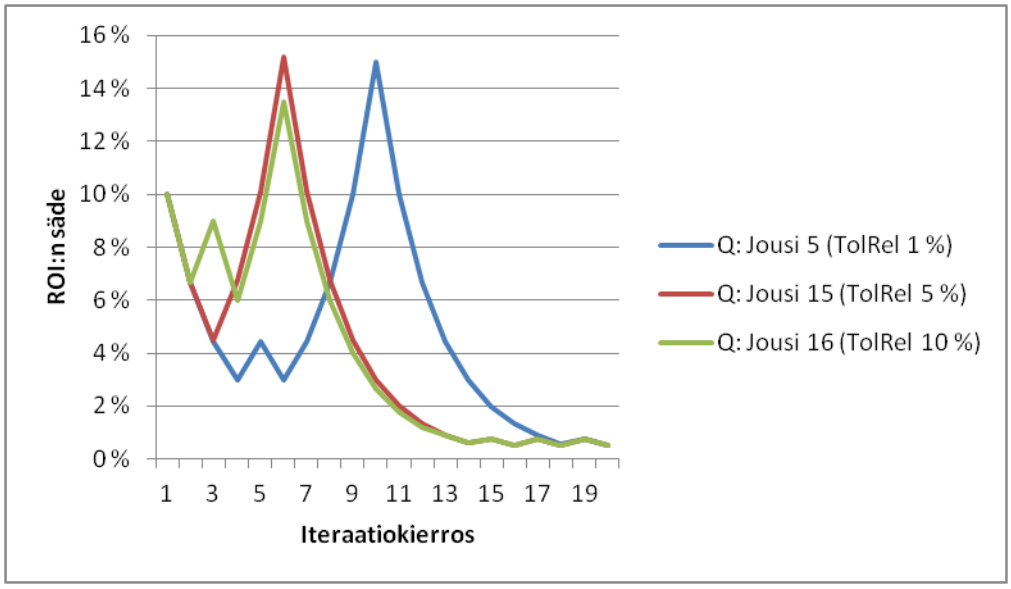
Seuraavaksi kokeiltiin kiinteää ROI:n sädettä, jotta nähdään paremmin approksimoidun mallin tarkkuus. ROI:n säteen kasvattaminen nopeutti suppenemista huomattavasti. Suurella ROI:n säteen arvolla päästiin muutamalla iteraatiolla lähelle optimia, mutta se aiheutti vasteeseen värähtelyä. Kvadraattinen malli oli selvästi tarkempi, kun ROI:n sädettä kasvatettiin, mutta 50 % oli sillekin liikaa ja joudutaan käyttämään epäkäypien pisteiden takia Interior point -algoritmia useassa iteraatiossa. Eli näissä iteraatioissa ei rikottu pelkästään virhetoleranssia, vaan kävi niin, että käypä joukko oli tyhjä.

Sitten kokeiltiin eri arvoja tavoitevirheelle. Tuloksista huomataan, että sen nostamisella ei ollut vaikutusta lineaariseen ja puhtaasti kvadraattiseen malliin. Kvadraattisen mallin kanssa suppeni huomattavasti paremmin, kun tavoitevirhettä nostettiin.

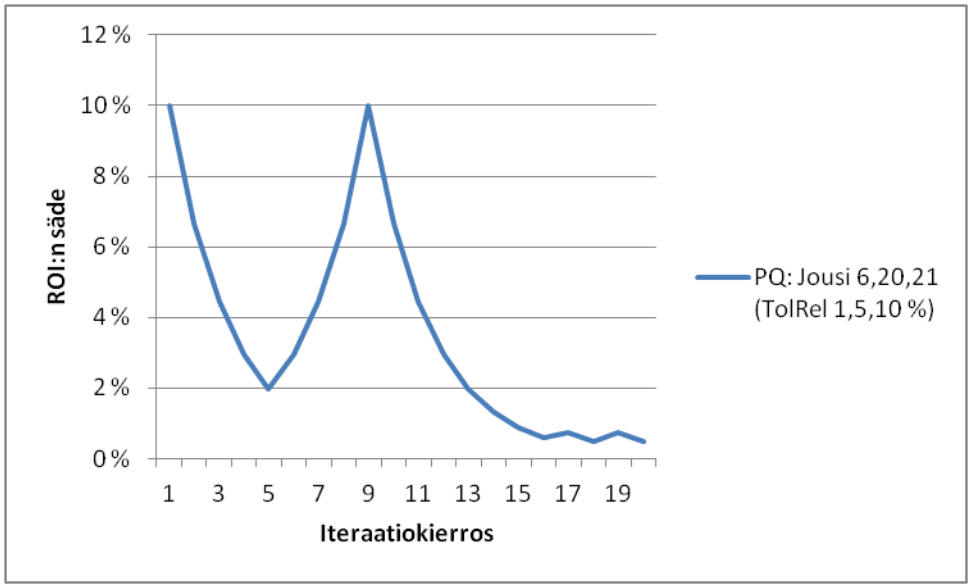
Kuvista 18-20 nähdään, miten ROI:n säde muuttuu iteraatiokierrosten aikana eri tavoitevirheellä. Ensimmäinen piste kuvaajassa on alkupiste, jonka jälkeen ROI:n säde lasketaan aina seuraavaa iteraatiokierrosta varten. Erittäin epälineaaristen funktioiden takia virhettä on kaikilla malleilla aika paljon, jolloin ROI:n sädettä pienennetään vaikka tavoitevirhe olisi suurempi. Lineaarisella ja puhtaasti kvadraattisella mallilla tavoitevirheen kasvattaminen ei siksi vaikuttanut ROI:n säteeseen. Kvadraattisella mallilla suurempi tavoitevirhe kasvattaa aikaisemmilla iteraatiokierroksilla ROI:n sädettä. Selvää vaikutusta tavoitevirheen kasvattamisella ei tässä tehtävässä kuitenkaan ollut. Lastiluukun optimoinnissa luvussa 5.3.1 tavoitevirheen kasvattamisen vaikutus nähdään paremmin.



Kuva 18 *Kierrejousitehtävän ROI:n säteen muuttuminen iteraatiokierrosten aikana (Lineaarinen malli).*



Kuva 19 *Kierrejousitehtävän ROI:n säteen muuttuminen iteraatiokierrosten aikana (Kvadraattinen malli).*



Kuva 20 *Kierrejousitehtävän ROI:n säteen muuttuminen iteraatiokierrosten aikana (Puhtaasti kvadraattinen malli).*

Viimeisissä ajoissa lähdettiin liikkeelle epäkäyvästä pisteestä käyttämällä 10 tai 50 % kiinteää ROI:n sädettä. Interior point on kokemuksen mukaan ainoa käytetyistä algoritmeista, joka selviää epäkäyvästä aloituspisteestä, johon sitten vaihdetaan tarvittaessa. Interior pointin palauttama piste ei kuitenkaan välttämättä ole edes mallille käypä, kuten edellä Optimointi-luvun esimerkissä havainnollistettiin. Tuloksista huomataan, että 10 % ROI:n säteellä ei löydetä käypiä pisteitä. Vasteet värähtelevät erittäin voimakkaasti. Kvadraattisella mallilla ja 50 % säteellä löydetään 2 käypää pistettä. Ison ROI:n ansiosta päästään lähelle käypää aluetta ja kvadraattisella mallilla päästiin käyvälle alueelle.

Seuraavaksi on analysoitu mallin ja algoritmin vaikutusta optimoinnin tarkkuuteen ja tehokkuuteen. Taulukosta 11 nähdään myös suunnittelumuuttujien arvot optimissa.

Taulukko 11 *Kierrejousen optimoinnin tulokset (mallien ja algoritmien vertailu).*

Tunniste	Jousi 1	Jousi 2	Jousi 3	Jousi 4	Jousi 5	Jousi 6
Algoritmi	ML IP	ML SX	ML AS	Gurobi	Scip	Scip
Malli	L	L	L	L	Q	PQ
Suunnittelu-muuttujat	8.815196 0.407957 0.053738	8.204587 0.424260 0.054357	8.204587 0.424260 0.054357	8.204587 0.424260 0.054357	10.210455 0.376621 0.052503	7.814772 0.435666 0.054778
Tarkka arvo	0.012741	0.012792	0.012792	0.012792	**0.012676**	0.012831
Mallin antama arvo	0.012743	0.012794	0.012794	0.012794	0.012676	0.012829
ROI:n säde %	0.75	0.75	0.75	0.75	0.75	0.5
Residuaali	2e-6	2e-6	2e-6	2e-6	0	-1e-6
Virhe %	0.014118	0.014130	0.014130	0.014130	6e-6	0.009106
Rajoiteyht. keskim. virhe $\overline{\boldsymbol{gviol}}$	0.0000494	0.0000494	0.0000494	0.0000494	0.0000081	0.0000453
Käypä	kyllä	kyllä	kyllä	kyllä	kyllä	kyllä
Käypien pis-teiden lkm	22	23	23	23	42	33
Iteraatio	50/50	50/50	50/50	50/50	44/45	49/50
Aika	223 s	296 s	219 s	219 s	215 s	227 s

Taulukosta 11 huomataan, että optimipisteessä tarkasteltava aliavaruus on mennyt hyvin pieneksi, laskentakoodissa asetettuun alarajaan asti ja malli on optimissa tarkka. Tämä huomataan myös pienestä suhteellisesta virheestä ja residuaalista. ROI:n säteen alarajaksi on koodiin asetettu 0.5 %. Iteraatioraportteja tarkemmin tutkimalla huomattiin, että ROI:n säde vaihteli 0.5 % tuntumassa. Lähellä optimia oltiin taulukon 11 ajoissa noin 15 kierroksen jälkeen. Tämän jälkeen vaste jäi värähtelemään optimin lähelle. Tässä virhetoleranssin rikkovia epäkäypiä pisteitä tuli lineaariselle mallille, juuri tuosta alarajan herätteestä. Alarajalla ROI:n sädettä nostetaan aika reilusti suhteessa edelliseen arvoon.

Algoritmien tarkemmassa vertailussa huomataan, että ainoastaan Interior point -algoritmin tulos poikkesi hieman muiden lineaaristen algoritmien tuloksista. Tämä johtunee siitä, että sisäpistemenetelmä eroaa näistä muista menetelmistä merkittävästi. Muut käyttävät aktiivijoukkomenetelmiin kuuluvia menetelmiä.

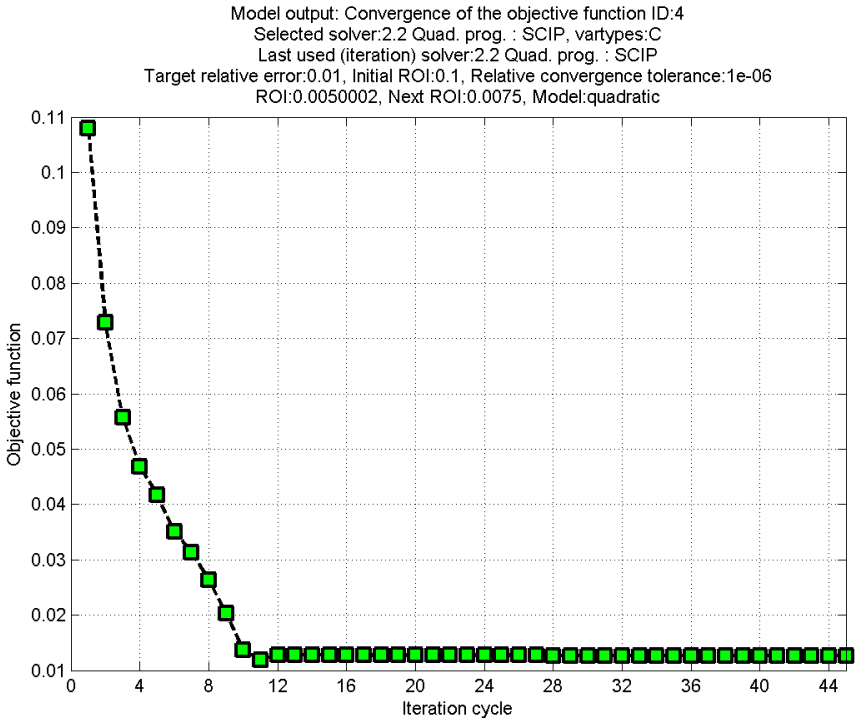
Liitteestä 13 nähdään kuuden ensimmäisen ajon rajoitefunktioiden suppeneminen. Kuvaajista huomataan, että rajoitefunktioissa on enemmän virhettä malliin nähden kuin kohdefunktiolla. Rajoitefunktioiden virhe vaikuttaa myös ROI:n säteeseen, koska sen laskemisessa käytetään kaikkien vasteiden suurinta suhteellista virhettä. Muuttujien kuvaajista liitteessä 14 nähdään hyvin miten isompi ROI vaikuttaa niihin. Etenkin lineaarisen mallin kanssa suunnittelumuuttujat muuttuvat rajusti. Adaptiivisen ROI:n säteen kanssa saadaan tasaisempi suppeneminen aikaan.

Tulosten perusteella kvadraattinen optimointi oli selvästi tarkempi kuin lineaarinen, mutta suppeni hieman hitaammin ensimmäisillä iteraatioilla. Kvadraattinen malli lopetti iteroinnin asetettuun tiukkaan lopetusehtoon. Puhtaasti kvadraattisella mallilla jäätiin vielä huonompaan tulokseen kuin lineaarisella, mikä johtunee todellisissa vasteissa olevista ristikkäisvaikutuksista. Taulukossa 12 vertaillaan vielä työkalulla saatua parasta ratkaisua kirjallisuudesta saatuun.

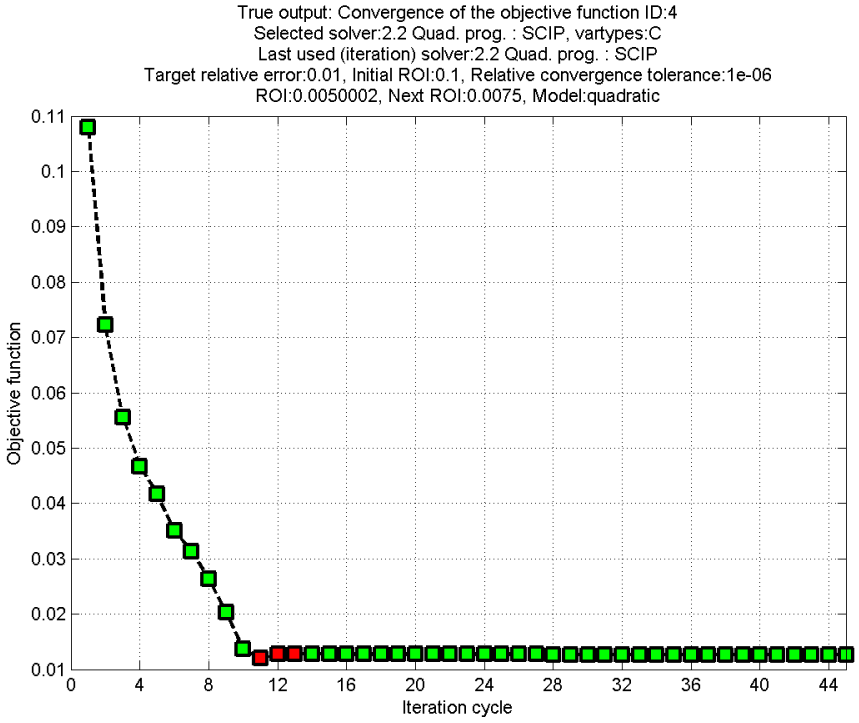
Taulukko 12 *Kierrejousen optimoinnille kirjallisuudesta löydetyt parhaat tulokset verrattuna parhaaseen laskentatyökalulla saatuun (Jousi 28, TolCon=1e-4).*

	Kirjallisuus (Lemonge et al. 2010)	Scip	Laskentatyökalu (paras TolCon sisällä) (iter 118/200)	Laskentatyökalu (pienin virhe) (iter 36/200)
Rajoiteyht. keskim. virhe	Ei tietoa	0	0.00002	0
Rajoiteyht. max. virhe	Ei tietoa	0	0.00005	0
Suunnittelu-muuttujat	11.661192 0.350530 0.051431	11.288963 0.356718 0.051689	11.256010 0.357264 0.051711	9.776533 0.385608 0.052862
Arvo	0,012666	0.012665	**0.012664**	0.012690

Tulos hieman parani kun nostettiin maksimi-iteraatioiden määrää. Tällöin päästiin todella hyvään tulokseen, mutta iteraatioiden määrä on hyvin suuri. Lähelle optimia päästiin kuitenkin jo 11 kierroksen jälkeen. Kuvista 21-22 nähdään työkalun piirtämät kuvaajat tarkemmin. Kuvista nähdään, että optimointi kvadraattisen mallin kanssa oletusasetuksilla on toiminut hyvin. Virhetoleranssin ylittäneitä pisteitä on vain kolme.

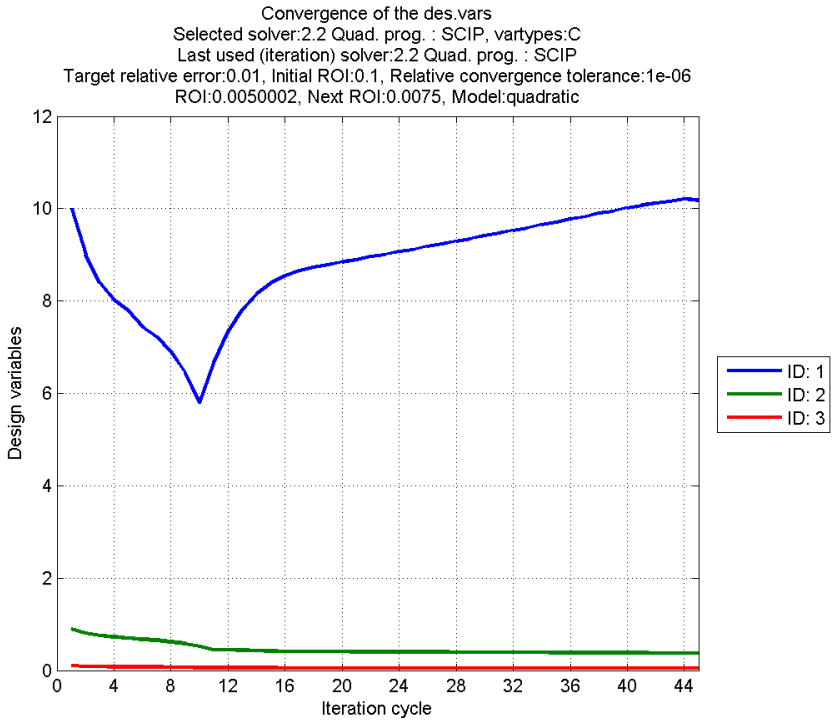


Kuva 21 *Jousi 5, kohdefunktion suppeneminen (mallin antama vaste).*



Kuva 22 *Jousi 5, kohdefunktion suppeneminen (todellinen vaste).*

Kuvasta 23 nähdään vielä muuttujien suppeneminen iteraatiokierrosten aikana. Kvadraattisella mallilla ei ilmene värähtelyä.



Kuva 23 *Jousi 5, muuttujien suppeneminen.*

Yhteenvetona kierrejousitehtävän optimoinnista voidaan sanoa, että erittäin epälineaarisissa tehtävissä lineaarisella mallilla kannattaa käyttää korkeintaan 10 % kiinteää ROI:n sädettä, kun taas kvadraattisella mallilla pystyttiin 20 % ROI:n säteelläkin approksimoimaan melko tarkasti. Suurella aliavaruuden koolla voidaan päästä muutamalla iteraatiolla lähelle optimia, mutta käypää pistettä ei välttämättä löydy. Adaptiivisella ROI:n säteellä päästiin kuitenkin kaikilla malleilla ja algoritmeilla tässä hyvään tulokseen. Tässä tehtävässä adaptiivisella ROI:n säteellä tulos hieman parani koko ajan iteraatioiden aikana, jolloin paras piste oli usein viimeinen piste. Kiinteällä ROI:n säteellä, näin ei yleensä ollut.

5.2.2 Hitsattu palkki

Seuraavan tehtävän tavoitteena on minimoida palkin kustannuksia seuraavalla kohdefunktiolla ja rajoitteilla (Kalyanmoy 2000)

$$
\begin{aligned}
C(h,l,t,b) &= 1.10471h^2l + 0.04811tb(14.0+l) \\
g_1(\tau) &= 13600 - \tau \geq 0 \\
g_2(\sigma) &= 30000 - \sigma \geq 0 \\
g_3(b,h) &= b - h \geq 0 \\
g_4(P_c) &= P_c - 6000 \geq 0 \\
g_5(\delta) &= 0.25 - \delta \geq 0 \\
0.125 &\leq h \leq 10 \\
0.1 &\leq l,t,b \leq 10
\end{aligned}
\tag{43}
$$

missä

$$\tau = \sqrt{(\tau')^2 + (\tau'')^2 + \frac{l\tau'\tau''}{\alpha}}$$

$$\tau' = \frac{6000}{\sqrt{2}hl}$$

$$\tau'' = \frac{6000(14 + 0.5l)\alpha}{2(0.707hl\left(\frac{l^2}{12} + 0.25(h+t)^2\right))}$$

$$\alpha = \sqrt{0.25(l^2 + (h+t)^2)}, \qquad \sigma = \frac{504000}{t^2 b}$$

$$P_c = 64746.022(1 - 0.0282346t)tb^3$$

$$\delta = \frac{2.1952}{t^3 b}.$$

Kaikkia muuttujia pidetään tässäkin tehtävässä jatkuvina. Taulukoissa 13-15 esitetään asetukset ja tulokset kuten edellisessä tehtävässä. Tehtäviin asetettiin kaikkiin kiinteä ROI:n säde, koska adaptiivisella säteellä jäätiin selvästi huonompaan tulokseen.

Taulukko 13 *Hitsatun palkin optimoinnissa käytetyt yleiset asetukset ja lähtötiedot.*

Faktoreiden lukumäärä ja tyypit	4 jatkuvaa muuttujaa
Rajoitusyhtälöiden lukumäärä	5
Minimoitava vaste $C(\mathbf{x})$	Kustannus
Aloituspiste $\mathbf{x}_0$	(3, 3, 3, 3)
Aloituspiste käypä	kyllä
Arvot aloituspisteessä	$C(\mathbf{x}_0)$ = 37.188 $g_1(\mathbf{x}_0)$ = -10842.767378 $g_2(\mathbf{x}_0)$ = -11333.333333 $g_3(\mathbf{x}_0)$ = 0 $g_4(\mathbf{x}_0)$ = -4794204.820039 $g_5(\mathbf{x}_0)$ = -0.222899
Rajoitusyhtälöiden oikea puoli	**0**
Globaalit rajat	$x_1 \in [0.125, 10]$, $x_2, x_3, x_4 \in [0.1, 10]$
Lopetusehto (konvergenssi)	1e-6 = 0,0001 %
Maksimi iteraatiot	50
Aloitus ROI (säde)	10 %
Adaptiivinen ROI:n säde	ei
Suunnittelumuuttujien tarkkuus	1e-12
Tavoitevirhe (TolRel)	1 % (ei vaikutusta kun ROI:n säde on kiinteä)
TolCon	1e-4
Techila käytössä	ei

Taulukko 14 *Hitsatun palkin optimointiajot.*

Tunniste	M	Algoritmi	Kuvaus	C	Käypä k/e (virhe)	Aika / s	Iter
HP 1	L	ML IP	Katso taulukko 13	2.509901	k (0)	260	24/50
HP 2	L	ML SX	Katso taulukko 13	2.500676	k (0)	258	24/50
HP 3	L	ML AC	Katso taulukko 13	2.500676	k (0)	258	24/50
HP 4	L	Gurobi	Katso taulukko 13	2.500676	k (0)	257	24/50
HP 5	Q	Scip	Katso taulukko 13	2.381169	k (0)	296	26/50
HP 6	PQ	Scip	Katso taulukko 13	2.389579	k (0)	280	45/50
HP 7	Q	Scip	HP 5, max iter = 200, convtol = 1e-12.	2.381144	k (0)	1385	198/ 200

Taulukko 15 *Hitsatun palkin optimoinnin tulokset (algoritmien vertailu).*

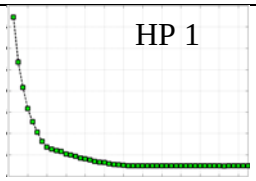
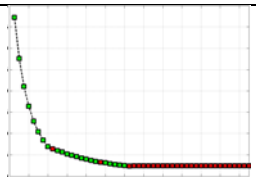
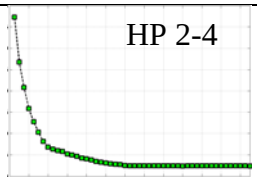
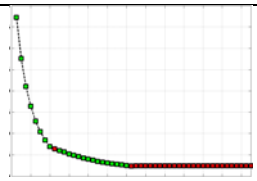
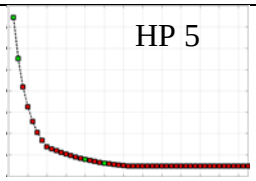
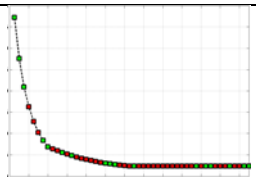
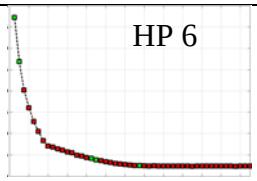
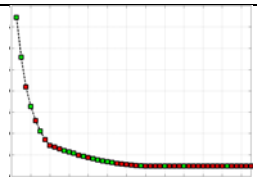
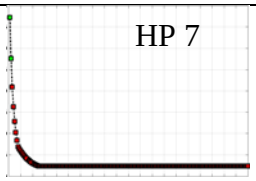
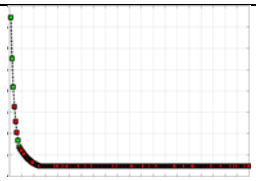
Tunniste	HP 1	HP 2	HP 3	HP 4	HP 5	HP 6
Algoritmi	ML IP	ML SX	ML AS	Gurobi	Scip	Scip
Malli	L	L	L	L	Q	PQ
Suunnittelu-muuttujat	0.265888 5.806749 8.116333 0.265888	0.265888 5.773678 8.103649 0.265888	0.265888 5.773678 8.103649 0.265888	0.265888 5.773678 8.103649 0.265888	0.244369 6.218452 8.291740 0.244369	0.244381 6.262137 8.296472 0.244381
Tarkka arvo	2.509901	2.500676	2.500676	2.500676	**2.381169**	2.389579
Mallin antama arvo	2.543296	2.517595	2.517595	2.517595	2.381170	2.401914
ROI:n säde %	10	10	10	10	10	10
Residuaali	0.033395	0.016919	0.016919	0.016919	0.000001	0.012335
Virhe %	1.330534	0.676568	0.676568	0.676568	0.000051	0.516219
Rajoiteyht. keskim. virhe	0	0	0	0	0	0
Käypä	kyllä	kyllä	kyllä	kyllä	kyllä	kyllä
Käypien pis-teiden lkm	22	23	23	23	17	17
Iteraatio	24/50	24/50	24/50	24/50	26/50	45/50
Aika	260 s	258 s	258 s	257 s	296 s	280 s

Jälleen huomataan, että lineaaristen tehtävien aktiivijoukkomenetelmiä käyttäville algoritmeille tuli täsmälleen samat tulokset. Interior pointilla saatu poikkeaa niistä vain hieman. Tuloksista nähdään, että kaikissa tehtävissä paras ratkaisu oli täysin käypä eli ei rikottu lainkaan asetettua virhetoleranssia, vaikka ROI:n säde oli 10 %. Malli approksimoi hyvin tällä optimipisteen iteraatiolla vastetta. Tämä nähdään myös pienestä suhteellisesta virheestä ja residuaalista. Kvadraattinen malli on todella tarkka optimipisteessä,

sillä suhteellinen virhe on vain 5e-5 %. Kvadraattiset mallit antoivat tässä hieman paremman tuloksen kuin lineaarinen.

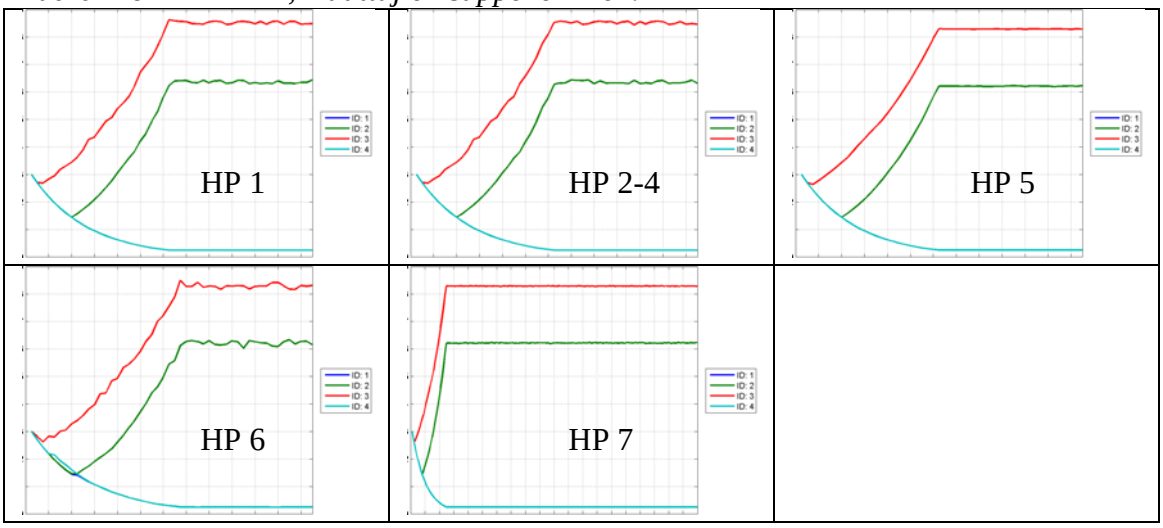
Taulukosta 16 nähdään hitsatun palkin kohdefunktioiden kuvaajat eri ajoissa. Taulukosta 17 nähdään muuttujien kuvaajat. Vasteiden vertailussa huomataan, ettei suppenemisessa ole juurikaan eroa eri tehtävien kesken. Kvadraattisen mallin kanssa epäkäypiä pisteitä on enemmän. Myös mallin antamia vasteita on kvadraattisissa tehtävissä merkitty epäkäyväksi. Tässä kävi niin kuin luvussa 4.1.3 kerrottiin. Eli tämä johtuu siitä, että Scip-algoritmi on sallinut tehtävälle enemmän virhettä verrattuna työkaluun annettuun virhetoleranssiin. Minkään tehtävän eri iteraatioilla optimoitava malli ei mennyt muuten epäkäyväksi, kuin rikkomalla asetettua virhetoleranssia.

Taulukko 16 *HP 1-7, kohdefunktioiden suppenemisen vertailu.*

Malli	Tarkka	Malli	Tarkka
HP 1 		HP 2-4 	
HP 5 		HP 6 	
HP 7 			

Muuttujien kuvaajista nähdään selvemmin eroa tehtävien välillä. Kvadraattisella mallilla muuttujat suppenevat tasaisesti, kun taas lineaarisella ja puhtaasti kvadraattisella mallilla näkyy pientä värähtelyä.

Taulukko 17 *HP 1-7, muuttujien suppeneminen.*



Taulukossa 18 vertaillaan vielä työkalulla saatua parasta ratkaisua kirjallisuudesta saatuun. Kvadraattinen tehtävä ajettiin vielä uudestaan nostamalla iteraatioiden määrää, jolloin tulos parani hieman edellisestä ajosta. Laskentatyökalulla löydettiin todella hyvä optimipiste, jossa ei rikottu asetettua virhetoleranssia lainkaan. Scipin suoraan laskema optimi taas rikkoo jonkin verran. Tämä selittää myös mallin antamien vasteiden epäkäypyyden kvadraattisissa ajoissa. Asetettu TolCon-toleranssi oli liian tiukka tähän tehtävään, koska rajoitefunktioiden vasteet ovat todella suuria.

Taulukko 18 *Hitsatun palkin optimoinnille kirjallisuudesta löydetyt parhaat tulokset verrattuna parhaaseen laskentatyökalulla saatuun (HP 7, TolCon=1e-4).*

	Kirjallisuus (Lemonge et al. 2010)	Scip	Laskentatyökalu (paras TolCon sisällä) (iter 198/200)
Rajoiteyht. keskim. virhe	Ei tietoa	4e-4	0
Rajoiteyht. max virhe	Ei tietoa	3e-4	0
Suunnittelu-muuttujat	0.244386 6.218304 8.291165 0.244388	0.264710 5.852359 7.966534 0.264710	0.244369 6.218620 8.291517 0.244369
Arvo	2.38122	2.46715	**2.38114**

Kaikki mallit ja algoritmit toimivat hyvin tässä tehtävässä vaikka käytettiin kiinteää ROI:n sädettä. Suuremmalla ROI:n säteellä jäi kokeilematta, että kuinka hyvään tulokseen sillä olisi voinut päästä. Tässä tehtävässä 10 % ROI:n säteellä kohdefunktio suppeni kuitenkin melko hitaasti, koska vasta 25 iteraation jälkeen oltiin lähellä optimipistet-

tä. Adaptiivista ROI:n sädettä koetettiin tässä tehtävässä ensin, mutta sillä kohdefunktio suppeni erittäin hitaasti kaikilla malleilla.

5.2.3 Golinskin nopeudenalennin

Nopeudenalentimen kohdefunktio ja rajoitteet ovat (Lemonge et al. 2010)

$$
\begin{aligned}
W(\mathbf{x}) &= 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\
&\quad -1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) \\
&\quad +0.7854(x_4x_6^2 + x_5x_7^2) \\
g_1(\mathbf{x}) &= 27x_1^{-1}x_2^{-2}x_3^{-1} \leq 1 \\
g_2(\mathbf{x}) &= 397.5x_1^{-1}x_2^{-2}x_3^{-2} \leq 1 \\
g_3(\mathbf{x}) &= 1.93x_2^{-1}x_3^{-1}x_4^3x_6^{-4} \leq 1 \\
g_4(\mathbf{x}) &= 1.93x_2^{-1}x_3^{-1}x_5^3x_7^{-4} \leq 1 \\
g_5(\mathbf{x}) &= \frac{1}{0.1x_6^3}\left[\left(\frac{745x_4}{x_2x_3}\right)^2 + (16.9)10^6\right]^{0.5} \leq 1100 \\
g_6(\mathbf{x}) &= \frac{1}{0.1x_7^3}\left[\left(\frac{745x_5}{x_2x_3}\right)^2 + (157.5)10^6\right]^{0.5} \leq 850 \\
g_7(\mathbf{x}) &= x_2x_3 \leq 40 \\
g_8(\mathbf{x}) &= x_1/x_2 \geq 5 \\
g_9(\mathbf{x}) &= x_1/x_2 \leq 12 \\
g_{10}(\mathbf{x}) &= (1.5x_6 + 1.9)x_4^{-1} \leq 1 \\
g_{11}(\mathbf{x}) &= (1.1x_7 + 1.9)x_5^{-1} \leq 1
\end{aligned}
\tag{44}
$$

Kaikki muuttujat ovat tässä jatkuvia ja tehtäviin valittiin adaptiivinen ROI. Taulukoissa 19-21 esitetään asetukset ja tulokset.

Taulukko 19 *Nopeudenalentimen optimoinnissa käytetyt yleiset asetukset ja lähtötiedot.*

Faktoreiden lukumäärä ja tyypit	7 jatkuvaa muuttujaa
Rajoitusyhtälöiden lukumäärä	11
Minimoitava vaste $W(\mathbf{x})$	Massa
Aloituspiste $\mathbf{x}_0$	(4.35, 0.78, 27, 8.25, 8.25, 3.85, 5.45)
Aloituspiste käypä	kyllä
Arvot aloituspisteessä	$W(\mathbf{x}_0)$ = 7433.081694 $g_1(\mathbf{x}_0)$ = -0.622148 $g_2(\mathbf{x}_0)$ = -0.793970 $g_3(\mathbf{x}_0)$ = -0.765783 $g_4(\mathbf{x}_0)$ = -0.941672 $g_5(\mathbf{x}_0)$ = -377.807784 $g_6(\mathbf{x}_0)$ = -74.524342 $g_7(\mathbf{x}_0)$ = -18.940000 $g_8(\mathbf{x}_0)$ = -0.576923 $g_9(\mathbf{x}_0)$ = -6.423077 $g_{10}(\mathbf{x}_0)$ = -0.069697 $g_{11}(\mathbf{x}_0)$ = -0.043030
Rajoitusyhtälöiden oikea puoli	**0**
Globaalit rajat	$x_1 \in [2.6, 3.6]$, $x_2 \in [0.7, 0.8]$, $x_3 \in [17, 28]$, $x_4 \in [7.3, 8.3]$, $x_5 \in [7.8, 8.3]$, $x_6 \in [2.9, 3.9]$, $x_7 \in [5, 5.5]$
Lopetusehto (konvergenssi)	1e-6 = 0,0001 %
Maksimi iteraatiot	50
Aloitus ROI (säde)	10 %
Adaptiivinen ROI	kyllä
Suunnittelumuuttujien tarkkuus	1e-12
Tavoitevirhe (TolRel)	1 %
TolCon	1e-4
Techila käytössä	ei

Taulukko 20 *Nopeudenalentimen optimointiajot.*

Tun-niste	M	Algo-ritmi	Kuvaus	Massa	Käypä k/e (virhe)	Aika / s	Iter
SR 1	L	ML IP	Katso taulukko 19	2996.5241	k (1e-5)	513	45/50
SR 2	L	ML SX	Katso taulukko 19	2996.5245	k (1e-5)	517	25/50
SR 3	L	ML AC	Katso taulukko 19	2996.5245	k (1e-5)	508	25/50
SR 4	L	Gurobi	Katso taulukko 19	2996.5245	k (1e-5)	508	25/50
SR 5	Q	Scip	Katso taulukko 19	2996.3474	k (8e-5)	297	24/25
SR 6	PQ	Scip	Katso taulukko 19	2996.2262	k (5e-5)	545	36/50
SR 7	Q	Scip	SR 5, max iter. 200, ConvTol = 1e-12.	2996.3468	k (8e-5)	2553	190/ 200
SR 8	PQ	Scip	SR 6, max iter. 200, ConvTol = 1e-12.	**2996.2194**	k (5e-5)	2848	90/200

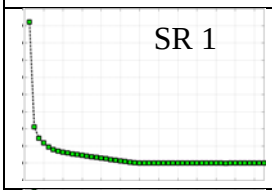
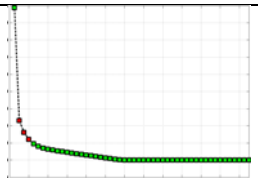
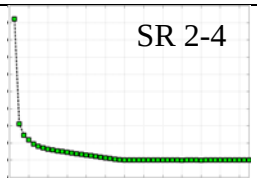
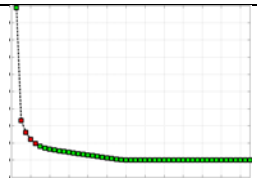
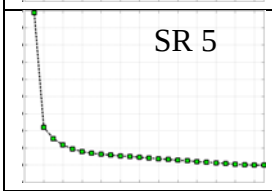
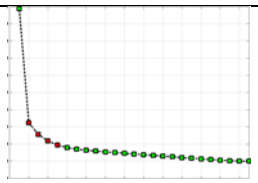
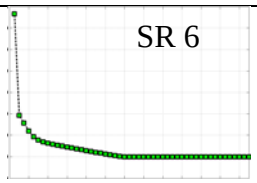
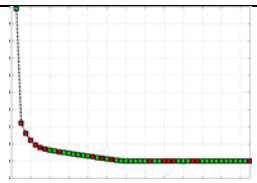
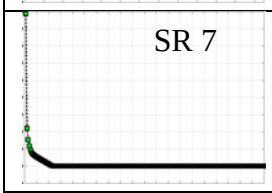
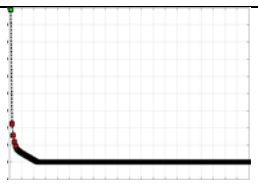
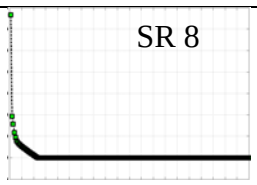
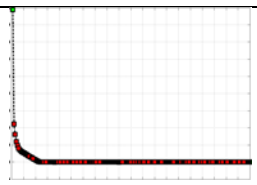
Taulukko 21 *Nopeudenalentimen optimoinnin tulokset (algoritmien vertailu).*

Tunniste	SR 1	SR 2	SR 3	SR 4	SR 5	SR 6
Algoritmi	ML IP	ML SX	ML AS	Gurobi	Scip	Scip
Malli	L	L	L	L	Q	PQ
Suunnittelu-muuttujat	3.499913 0.700000 17.000000 7.300000 7.800000 3.350380 5.286947	3.499912 0.700000 17.000000 7.300000 7.800000 3.350382 5.286948	3.499912 0.700000 17.000000 7.300000 7.800000 3.350382 5.286948	3.499912 0.700000 17.000000 7.300000 7.800000 3.350382 5.286948	3.500000 0.700000 17.000000 7.300000 7.800000 3.350214 5.286682	3.499684 0.700000 17.000000 7.300000 7.800000 3.350223 5.286683
Tarkka arvo	2996.5241	2996.5245	2996.5245	2996.5245	2996.3474	**2996.2262**
Mallin antama arvo	2996.4840	2996.3271	2996.3271	2996.3271	2996.3429	2995.1615
ROI:n säde %	0.5	0.5	0.5	0.5	0.75	0.75
Residuaali	-0.040140	-0.197378	-0.197378	-0.197378	-0.004418	-1.064766
Virhe %	0.001340	0.006587	0.006587	0.006587	0.000147	0.035537
Rajoiteyht. keskim. virhe	0.0000113	0.0000114	0.0000114	0.0000114	0.0000846	0.0000540
Käypä	kyllä	kyllä	kyllä	kyllä	kyllä	kyllä
Käypien pis-teiden lkm	47	46	46	46	21	33
Iteraatio	45/50	25/50	25/50	25/50	24/25	36/50
Aika	8 min, 33 s	8 min, 37 s	8 min, 28 s	8 min, 28 s	4 min, 57 s	9 min, 5 s

Tällä kertaa kaikkien lineaaristen tehtävien tulokset olivat samat, Interior point erosi vain hyvin vähän muista. Tuloksista huomataan, että kaikissa tehtävissä päästiin lähes samaan tulokseen. Kohdefunktion arvo pienenee radikaalisti yhdellä iteraatiolla kaikissa tehtävissä. Taulukosta 21 nähdään, että ROI:n säde on mennyt jälleen lähellä optimia todella pieneksi, asetettuun alarajaan asti. Lineaarisella mallilla ROI:n säde menee jo 9 kierroksen jälkeen alarajalle, johon se jäi vastaavasti muuttumaan kuin kierrejousitehtävässä. Tässä ROI:n säteen kasvattaminen alarajalla ei kuitenkaan aiheuttanut virhettä. Lineaarisella mallilla optimipisteen lähelle päästään vasta kierroksella 25, johtuen ROI:n säteen pienenemisestä. Kuitenkin adaptiivisen ROI:n ansiosta malli on iteraatioilla tarkka lukuun ottamatta muutamaa ensimmäistä pistettä joissa ROI on isompi. Löydetyissä parhaissa tuloksissa rikotaan hyvin vähän rajoitusyhtälöitä.

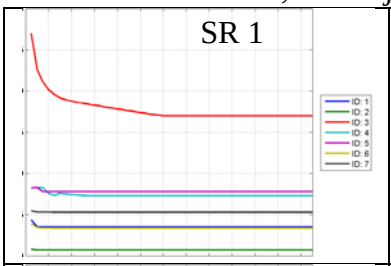
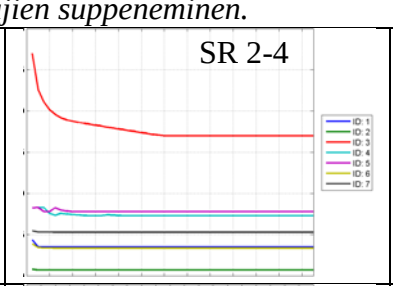
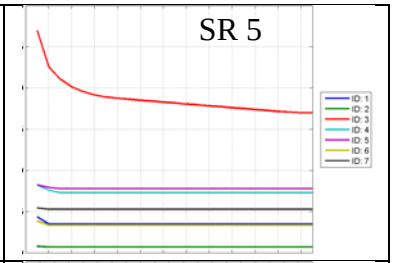
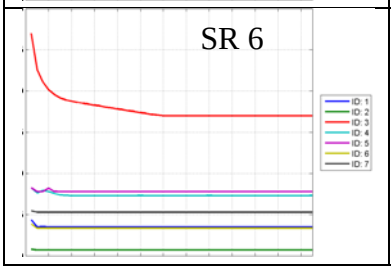
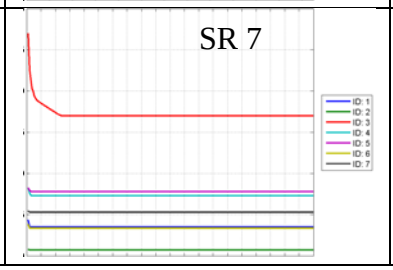
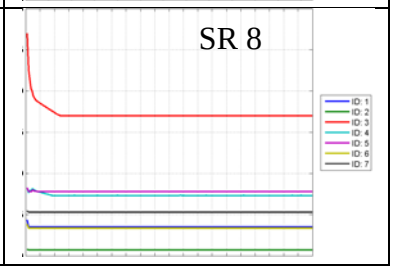
Myös kvadraattinen ja puhtaasti kvadraattinen malli jäi 24 kierroksen jälkeen ROI:n säteen alarajalle. Kvadraattinen malli lopetti tosin iteroinnin 25 kierroksen jälkeen. Puhtaasti kvadraattisella mallilla tuli kuten edellisestä taulukosta ja taulukon 22 kuvaajista nähdään epäkäypiä pisteitä aika paljon. Tämä johtunee hyvin epälineaarisen tehtävän muuttujien ristikkäisvaikutuksista. Kuvaajista nähdään, että eri mallien ja algoritmien kanssa kohdefunktio on supennut hyvin samalla tavalla.

Taulukko 22 *SR 1-8, kohdefunktioiden suppenemisen vertailu.*

Malli	Tarkka	Malli	Tarkka
			
			
			

Suunnittelumuuttujien suppenemisessakaan ei ole silminnähtävää eroa kuten taulukon 23 kuvaajista nähdään. Ajot SR 5 ja SR 7-8 näyttävät erilaiselta iteraatioiden eri määrän takia.

Taulukko 23 *SR 1-8, muuttujien suppeneminen.*

Kirjallisuudesta löytyvän tuloksen vertailua varten, ajettiin edellä saaduista paras eli puhtaasti kvadraattinen uudestaan nostamalla iteraatioiden määrää. Vertailun vuoksi ajettiin myös kvadraattinen uudestaan. Puhtaasti kvadraattisella löytyi hieman parempi, jonka tulokset nähdään taulukosta 24. Sallimalla pieni rajoitusyhtälöiden rikkominen päästiin hieman parempaan tulokseen kuin vertailtavana oleva tulos. Täysin käypä oli myös hyvä tulos. Laskemalla tehtävän suoraan Scip-algoritmilla päästiin vielä parempaan tulokseen, mutta siinä rikottiin hieman enemmän rajoitteita.

Taulukko 24 *Nopeudenalentimen optimoinnille kirjallisuudesta löydetyt parhaat tulokset verrattuna parhaaseen laskentatyökalulla saatuun (SR 8, TolCon=1e-4).*

	Kirjallisuus (Lemonge et al. 2010)	Scip	Laskentatyökalu (paras TolCon sisällä) (iter 90/200)	Laskentatyökalu (pienin virhe) (iter 131/200)
Rajoiteyht. keskim. virhe	Ei tietoa	Ei tietoa	0.000048668858	0
Rajoiteyht. max virhe	Ei tietoa	0.0038382	0.000476	0
Suunnittelu-muuttujat	3.5 0.7 17 7.3 7.8 3.350215 5.286683	3.5 0.7 17 7.3 7.8 3.350211 5.286547	3.499667 0.7 17 7.3 7.8 3.350223 5.286683	3.500011 0.7 17 7.3 7.8 3.350215 5.286684
Arvo	2996.3482	2996.2602	**2996.2194**	2996.3530

Tehtävä vaikutti etukäteen katsottuna erittäin haastavalta, mutta työkalulla päästiin adaptiivisen ROI:n ansioista todella tarkkaan tulokseen. Kiinteällä ROI:n säteellä mallit eivät olleet kovinkaan tarkkoja, joten päätettiin käyttää adaptiivista. Kiinteällä ROI:n säteellä päästiin kuitenkin muutamalla kierroksella lähelle optimipistettä. Tässä tehtävässä olisi ollut hyvä ensin hakea kiinteällä ROI:n säteellä optimipisteen lähelle, jonka jälkeen olisi voinut jatkaa adaptiivisella ROI:n säteellä ja hakea tarkemmin käypää pistettä. Kiinteän ROI:n säteen tuloksia ei tässä tarkemmin esitellä.

Koska huomattiin, että saatiin edellä samoja tuloksia lineaarisilla algoritmeilla, niin jatkossa testataan vain yhtä MATLABin lineaarista algoritmia. Jatkoon valittiin Interior point -algoritmi.

5.2.4 Kokonaislukutehtävät

Seuraavaksi esitellään kaksi yksinkertaista kokonaislukuoptimointitehtävää, joita on optimoitu kokonaislukutehtävänä Scip- ja Gurobi-algoritmeilla sekä jatkuvana tehtävänä MATLABin Interior point -algoritmilla, josta on haettu MATLABin binäärioptimoinnilla kokonaislukuoptimi. Tarkastelun kohteena on erityisesti diskreetin pisteen haku jatkuvan optimin lähiympäristöstä binäärioptimoinnilla (Heinonen & Pajunen 2011).

Ensimmäisessä kokonaislukuoptimointitehtävässä TP12 (Hock & Schittkowski 1981) on kvadraattiset kohde- ja rajoitusfunktiot.

$$f(\mathbf{x}) = 0.5x_1^2 + x_2^2 - x_1x_2 - 7x_1 - 7x_2$$
$$g_1(\mathbf{x}) = 25 - 4x_1^2 - x_2^2 \geq 0 \quad (45)$$

Tehtävässä muuttujien oletetaan olevan kokonaislukuja. Jatkuva optimi on hyvin lähellä kokonaislukuoptimia, riippuen mitä algoritmia käytetään. Tuloksista nähdään, että kvadraattisella mallilla löytyy sama optimi kaikissa ajoissa. Taulukoissa 25-26 on esitelty käytetyt asetukset ja ajojen tulokset. Tässä testiprobleemassa kokeiltiin vain lineaarista ja kvadraattista mallia.

Taulukko 25 *TP12-tehtävän optimoinnissa käytetyt yleiset asetukset ja lähtötiedot.*

Faktoreiden lukumäärä ja tyypit	2 kokonaislukumuuttujaa
Rajoitusyhtälöiden lukumäärä	1
Minimoitava vaste $f(\mathbf{x})$	Kustannus
Aloituspiste $\mathbf{x}_0$	Katso taulukko 26
Aloituspiste käypä	kyllä / ei, katso taulukko 26
Kohdefunktion arvo aloituspisteessä	Katso taulukko 26
Rajoitusyhtälöiden oikea puoli	**0**
Globaalit rajat	Rajoittamaton
Lopetusehto (konvergenssi)	1e-4 = 0.01 %
Maksimi iteraatiot	20
Aloitus ROI	10 %
Adaptiivinen ROI	kyllä
Suunnittelumuuttujien tarkkuus	1e-12 (ML IP + DH), muille 0.
Tavoitevirhe (TolRel)	1 %
TolCon	1e-4
Techila käytössä	ei

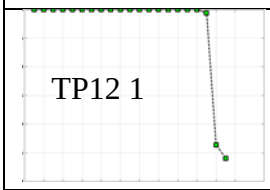
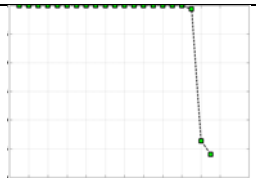
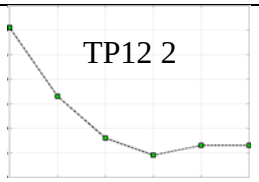
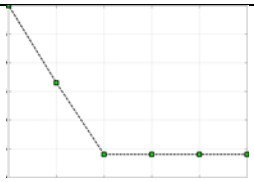
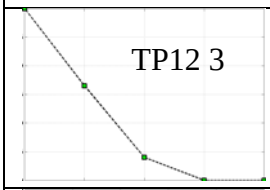
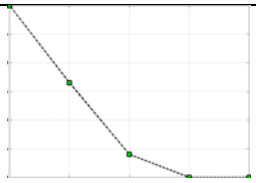
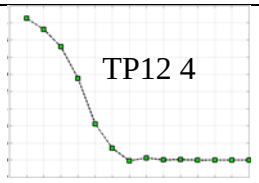
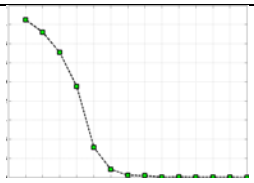
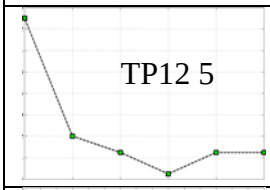
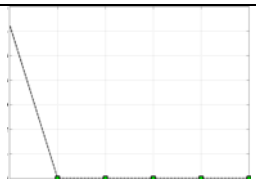
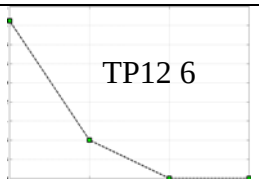
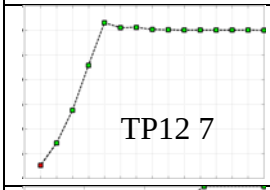
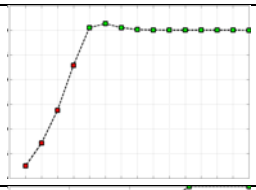
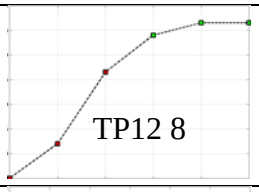
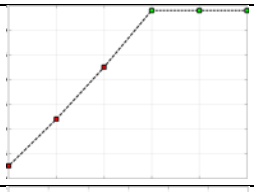
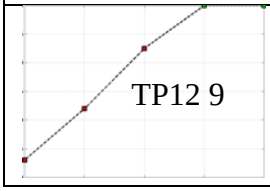
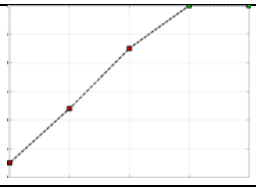
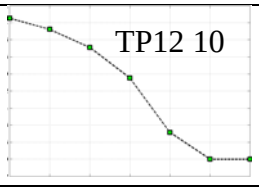
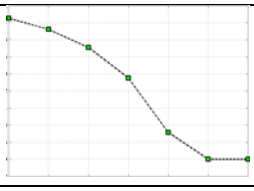
Tuloksista nähdään, että Gurobin kanssa jäädään kaikilla aloituspisteillä lokaaliin minimiin pisteeseen (2, 2). Interior point -algoritmilla diskreetin pisteen haun kanssa löydetään alkupisteestä (1, 1) ja epäkäyvästä alkupisteestä (5, 5) globaali optimi, mutta alkupisteellä (0, 0) jäädään lokaaliin minimiin. (0,0) on kirjallisuudessa annettu alkupiste. Laskentatyökalun Scip-algoritmilla päädytään kaikilla aloituspisteillä globaaliin optimiin. Epäkäyvästä pisteestä lähdettäessä käytetään Interior point -algoritmia myös Scipin ja Gurobin kanssa, kunnes päästään käyvälle alueelle. Kaikki tulokset ovat käypiä.

Taulukko 26 *Tulokset optimointiprobleemalle TP12.*

Ajo	Algoritmi	M	$\mathbf{x}_0$	$f(\mathbf{x}_0)$	$\mathbf{x}^*$	$f(\mathbf{x}^*)$	Iter
Kirjallisuus (Hock & Schittkowski 1981)	-	-	(0, 0)	0	(2,3)	-30	-
Scip	Scip	-	(0, 0)	0	(2,3)	-30	-
Scip	Scip	-	(1, 1)	-13.5	(2,3)	-30	-
Scip	Scip	-	(5, 5)	-57.5	(2,3)	-30	-
TP12 1 (Jatkuva)	ML IP	L	(0, 0)	0	(2.290157, 1.279691)	-23.659609	20/20
TP12 1	ML IP + DH	L	(0, 0)	0	(2,2)	-26	21/21
TP12 2	Gurobi	L	(0, 0)	0	(2,2)	-26	3/6
TP12 3	Scip	Q	(0, 0)	0	(2,3)	-30	4/5
TP12 4 (Jatkuva)	ML IP	L	(1, 1)	-13.5	(2.011775, 2.968024)	-29.996805	13/13
TP12 4	ML IP + DH	L	(1, 1)	-13.5	(2,3)	-30	14/14
TP12 5	Gurobi	L	(1, 1)	-13.5	(2,2)	-26	2/6
TP12 6	Scip	Q	(1, 1)	-13.5	(2,3)	-30	3/4
TP12 7 (Jatkuva)	ML IP	L	(5, 5)	-57.5	(1.998096, 3.004937)	-29.999541	14/14
TP12 7	ML IP + DH	L	(5, 5)	-57.5	(2,3)	-30	15/15
TP12 8	Gurobi	L	(5, 5)	-57.5	(2,2)	-26	4/6
TP12 9	Scip	Q	(5, 5)	-57.5	(2,3)	-30	4/5
TP12 10 (jatkuva) Katso liite 10	Scip	Q	(1, 1)	-13.5	(2,3)	-30	6/7

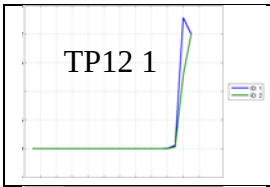
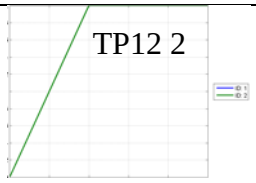
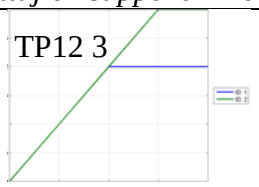
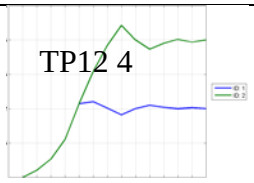
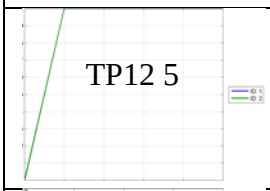
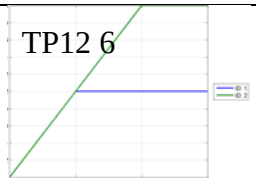
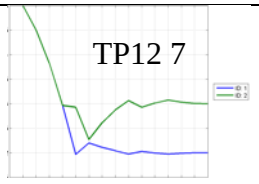
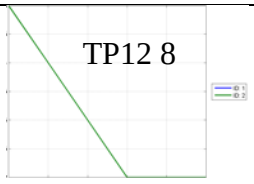
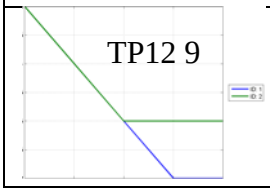
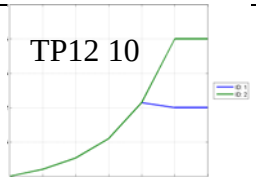
Taulukon 27 ensimmäisestä kuvaajasta nähdään kuinka hankala alkupiste (0,0) on tässä lineaariselle mallille ja Interior point -algoritmille. Vaikka malli on tarkka kaikilla iteraatiokierroksilla, joka nähdään työkalun tarkemmista tuloksista paremmin, niin silti lineaarinen malli vasta iteraatiolla 20 hypähtää parempaan arvoon. Suuri hyppy johtuu siitä, että ROI:n säde on siinä hyvin suuri. Tehtävälle oli asetettu iteraatioiden maksimimääräksi 20, joten siihen lopetettiin ja haettiin kokonaislukuoptimi binäärioptimoinnilla saadun pisteen (2.290157, 1.279691) ympäriltä. Binäärioptimointi palauttaa pisteen (2, 2), koska globaali optimi (2, 3) on sen tavoittamattomissa. Kyseinen menetelmä hakee tällä hetkellä kokonaislukuoptimia ottamalla rajoiksi jatkuvasta optimipisteestä katon (ceil) ja lattian (floor), jolloin tulee rajoiksi ensimmäiselle muuttujalle [2, 3] ja toiselle [1, 2]. Ainoastaan kvadraattisella mallilla kokonaislukutehtävänä päästiin alkupisteestä (0, 0) globaaliin optimiin. Koska todelliset funktiot ovat myös kvadraattiset, pystyy kvadraattinen metamalli mallintamaan sen täysin, mutta 10 % ROI:n alkusäteellä vaatii kuitenkin kolme iteraatiota, että löytää optimin. Suurella ROI:n säteellä, oltaisiin kvadraattisella mallilla löydetty optimi yhdellä kierroksella.

Taulukko 27 *TP12-tehtävän kohdefunktioiden suppeneminen.*

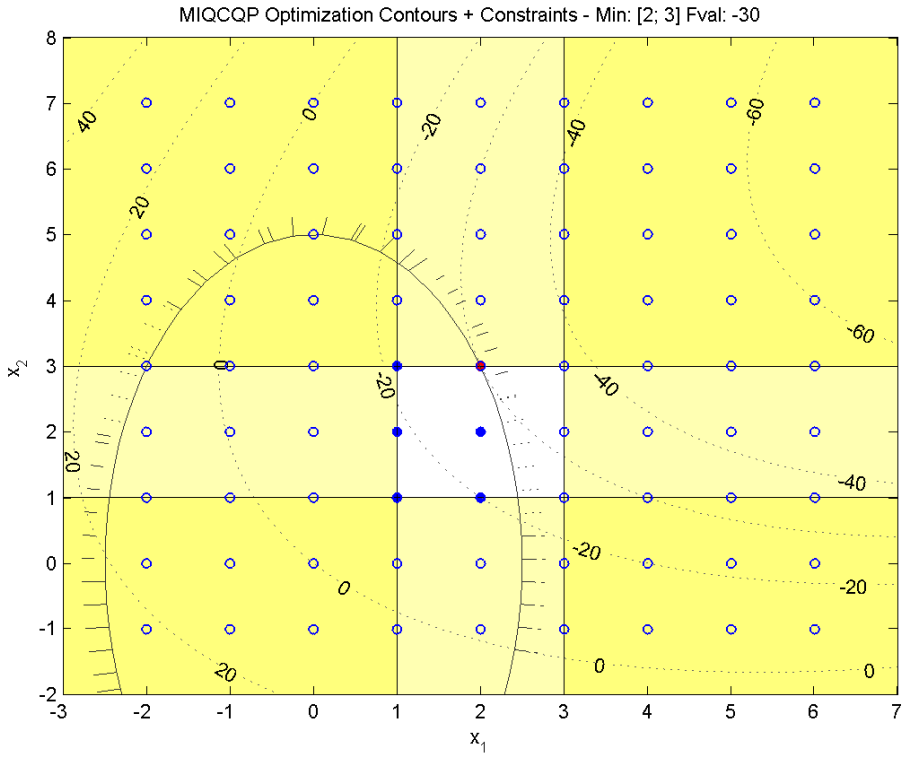
Malli	Tarkka	Malli	Tarkka
			
			
			
			
			

Taulukosta 28 nähdään miten muuttujat käyttäytyvät eri iteraatiokierroksilla. Mielenkiintoisin niistä on 7 ajo, missä lähdetään epäkäyvästä pisteestä jatkuvana optimoimaan ja lopuksi löydetään globaali kokonaislukuoptimi.

Taulukko 28 *TP12-tehtävän suunnittelumuuttujien suppeneminen.*

Scipin piirtämästä kuvasta 24 nähdään kokonaislukutehtävän viimeisen iteraatiokierroksen käypä joukko. Kuva on piirretty 6 ajosta. Käypiä kokonaislukupisteitä mahtuu käypään joukko kuusi. Liitteessä 10 on esimerkki, jossa havainnollistetaan TP12 10 -ajon avulla, miten tarkasteltava aliavaruus muuttuu iterointikierrosten aikana.



Kuva 24 *TP12 6, viimeisen iteraatiokierroksen käypä joukko (kokonaislukutehtävä).*

Toinen esimerkki on tapaus, jossa diskreetin pisteen haku ei löydä globaalia kokonaislukuoptimia millään testatuilla alkupisteillä. Seuraavaksi esitetään vain aloituspiste (1, 1). Tehtävä E3 (Loh & Papalambros 1991) on

$$\begin{aligned} f(\mathbf{x}) &= (x_1 - 8)^2 + (x_2 - 2)^2 \\ g_1(\mathbf{x}) &= 0.1x_1^2 - x_2 \leq 0 \\ g_2(\mathbf{x}) &= \frac{1}{3}x_1 + x_2 - 4.5 \leq 0. \end{aligned} \tag{46}$$

Taulukoissa 29-30 on esitelty käytetyt asetukset ja ajojen tulokset. Pyöristämällä jatkuva tulos saadaan (5, 3), joka on epäkäypä piste. Pyöristäminen ei siis ole yleensä kannattavaa. Interior point -algoritmilla diskreetin pisteen haun kanssa löytyy epäkäypä piste (5, 2). Edellä kuvatulla tavalla on saatu binäärioptimointia varten rajoiksi ensimmäiselle muuttujalle [5, 6] ja toiselle [2, 3]. Globaali optimi (4, 2) on siten binäärioptimoinnin tavoittamattomissa.

Taulukko 29 *E3-tehtävän optimoinnissa käytetyt yleiset asetukset ja lähtötiedot.*

Faktoreiden lukumäärä ja tyypit	2 kokonaislukumuuttujaa
Rajoitusyhtälöiden lukumäärä	2
Minimoitava vaste $f(\mathbf{x})$	Kustannus
Aloituspiste $\mathbf{x}_0$	(1, 1)
Aloituspiste käypä	kyllä
Arvot aloituspisteessä	$f(\mathbf{x}_0) = 50.000000$ $g_1(\mathbf{x}_0) = -0.900000$ $g_1(\mathbf{x}_0) = -3.166667$
Rajoitusyhtälöiden oikea puoli	**0**
Globaalit rajat	Rajoittamaton
Lopetusehto (konvergenssi)	1e-4 = 0.01 %
Maksimi iteraatiot	20
Aloitus ROI	10 %
Adaptiivinen ROI	Kyllä
Suunnittelumuuttujien tarkkuus	1e-12 (IP+DH), muille 0.
Tavoitevirhe (TolRel)	1 %
TolCon	1e-4
Techila käytössä	ei

Taulukko 30 *Tulokset optimointiprobleemalle E3.*

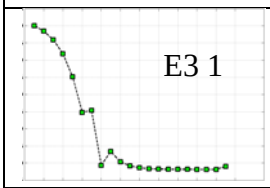
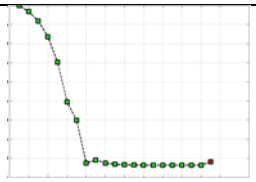
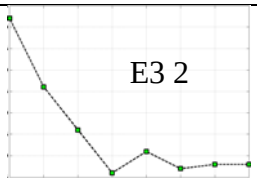
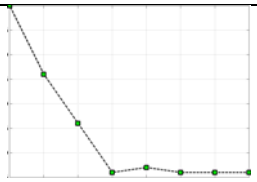
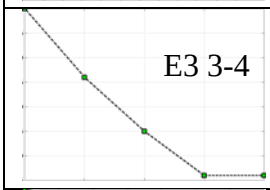
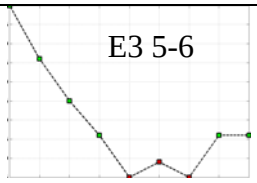
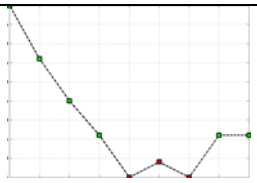
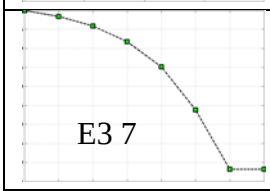
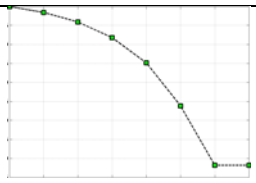
Tunniste	Algoritmi	M	$\mathbf{x}^*$	$f(\mathbf{x}^*)$	Käypä k/e (virhe)	Iter
Kirjallisuus		-	(4, 2)	16	-	-
Scip	Scip	-	(4, 2)	16	-	-
E3 1 (Jatkuva)	ML IP	L	(5.245368, 2.751544)	8.152813	k (0)	20/20
E3 1	ML IP + DH	L	(5, 2)	9	e (0.25)	21/21
E3 2	Gurobi	L	(4, 2)	16	k (0)	4/8
E3 3	Scip	Q	(4, 2)	16	k (0)	4/5
E3 4	Scip	PQ	(4, 2)	16	k (0)	4/5
E3 5	Gurobi	Q	(4, 2)	16	k (0)	4/9
E3 6	Gurobi	PQ	(4, 2)	16	k (0)	4/9
E3 7 (Jatkuva)	Scip	Q	(5.245481, 2.751507)	8.152139	k (1e-7)	8/8

Kaikissa muissa paitsi diskreetin pisteen haussa löydetään globaali optimi. Huomataan, että diskreetin pisteen haun tulos rikkoo ensimmäistä rajoitetta 0.5 verran eli reilusti. Tässä tehtävässä kokeiltiin myös Gurobia kvadraattisilla malleilla ja saatiin niillä hyvät tulokset, koska tehtävä pysyi iterointien aikana konveksina.

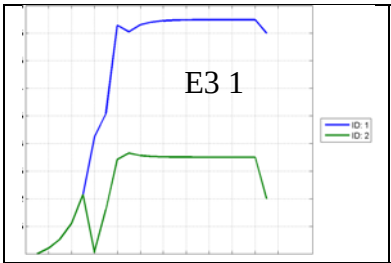
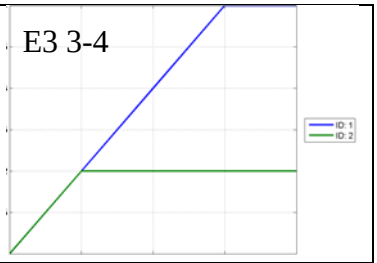
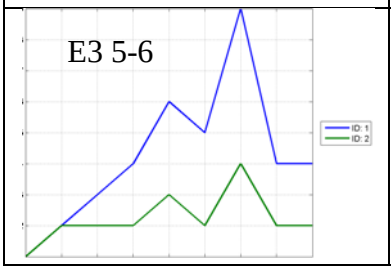
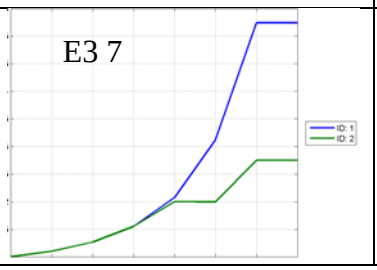
Taulukoista 31-32 nähdään, että kvadraattisilla malleilla kuvaajat ovat täsmälleen samat. Tämä johtuu siitä, että optimointitehtävässä ei ole ristitermejä, joten puhtaasti kvadraattinen on yhtä tarkka kuin kvadraattinen. Kvadraattinen optimointi Gurobilla

löytää optimin jo neljännellä iteraatiolla, mutta kohdefunktio ei suppene hyvin vaan siinä ilmenee värähtelyä. Tuloksia tarkemmin katsomalla huomataan, että Gurobin algoritmissa ja tuloksissa on epätarkkuutta. Scip-algoritmilla kvadraattinen optimointi sujuu jälleen hyvin.

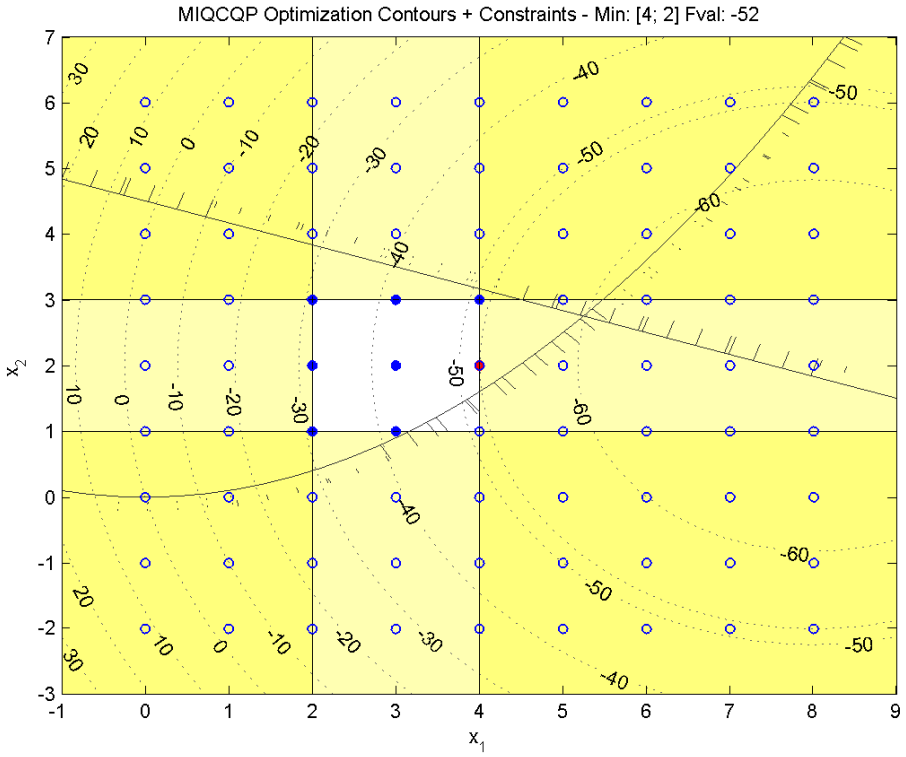
Taulukko 31 *E3-tehtävän kohdefunktioiden suppeneminen.*

Malli	Tarkka	Malli	Tarkka
			
			
			

Taulukko 32 *E3-tehtävän suunnittelumuuttujien suppeneminen.*

Scipin piirtämästä kuvasta 25 nähdään kokonaislukutehtävän viimeisen iteraatiokierroksen käypä joukko. Kuva on piirretty ajosta 3 . Kuvasta nähdään hyvin se, että pisteet (5,2) ja (5,3) ovat käyvän alueen ulkopuolella.



Kuva 25 *E3 3, viimeisen iteraatiokierroksen käypä joukko (kokonaislukutehtävä).*

Näillä kahdella kokonaislukutehtävällä nähdään, että käyttämällä diskreetin pisteen hakua jatkuvan optimin jälkeen, saattaa jäädä paremmat pisteet löytymättä. Jatkuvan pisteen pyöristäminen on vielä huonompi vaihtoehto. Binäärioptimointia voisi jatkokehityksessä kehittää siten, että se hakee jatkuvan optimin ympäriltä hieman suuremmasta pistejoukosta parasta kokonaislukuoptimia.

5.2.5 Painesäiliö

Seuraavilla painesäiliö- ja ulokepalkkitehtävillä on testattu luvussa 3.4 määriteltyjä malleja diskreeteille joukoille, jotka on implementoitu myös laskentatyökaluun. Laskentatyökalussa tehtäville, joissa on diskreettejä joukkoja, valitaan algoritmiksi ”Discrete sets: Gurobi/Scip”.

Minimoidaan painesäiliön tilavuutta seuraavilla rajoitteilla (Hsu et al. 2001)

$$
\begin{aligned}
f(\mathbf{x}) &= 0.6224x_1x_3x_4 + 1.778x_2x_3^2 \\
&\quad +3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
g_1(\mathbf{x}) &= -x_1 + 0.0193x_3 \leq 0 \\
g_2(\mathbf{x}) &= -x_2 + 0.00954x_3 \leq 0 \\
g_3(\mathbf{x}) &= -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0 \\
g_4(\mathbf{x}) &= x_4 - 240 \leq 0
\end{aligned}
\tag{47}
$$

missä x_1 on säiliön rungon ja x_2 kannen paksuus. Säiliön sylinteriosan sisähalkaisija on x_3 ja pituus on x_4. Kaikki muuttujat valitaan diskreeteistä joukoista, jotka nähdään taulukosta 33.

Taulukko 33 *Painesäiliön optimoinnissa käytetyt yleiset asetukset ja lähtötiedot.*

Faktoreiden lukumäärä ja tyypit	4 diskreettiä muuttujaa
Rajoitusyhtälöiden lukumäärä	4
Minimoitava vaste $f(\mathbf{x})$	Tilavuus
Aloituspiste $\mathbf{x}_0$	(1.25, 0.625, 50, 120)
Aloituspiste käypä	kyllä
Arvot aloituspisteessä	$f(\mathbf{x}_0) = 9589.925000$ $g_1(\mathbf{x}_0) = -0.285000$ $g_2(\mathbf{x}_0) = -0.148000$ $g_3(\mathbf{x}_0) = -170076.571675$ $g_4(\mathbf{x}_0) = -120.000000$
Rajoitusyhtälöiden oikea puoli	**0**
Globaalit rajat	$x_1 \in [0.625, 1.25]$ $x_2 \in [0, 0.625]$ $x_3 \in [45, 50]$ $x_4 \in [100, 120]$
Diskreetit joukot	$x_1 \in \{0.625, 0.6875, 0.75, 0.8125, 0.875, 0.9375, 1, 1.0625, 1.125, 1.1875, 1.25\}$ $x_2 \in \{0.0, 0.0625, 0.125, 0.1875, 0.25, 0.3125, 0.375, 0.4375, 0.5, 0.5625, 0.625\}$ $x_3 \in \{45, 45.5, 46, 46.5, 47, 47.5, 48, 48.5, 49, 49.5, 50\}$ $x_4 \in 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120\}$
Lopetusehto (konvergenssi)	1e-4 = 0.01 %
Maksimi iteraatiot	10
Aloitus ROI	5 %
Adaptiivinen ROI	kyllä
Suunnittelumuuttujien tarkkuus	1e-4, 1e-4, 1e-1, 0
Tavoitevirhe (TolRel)	1 %
TolCon	1e-2
Techila käytössä	ei

Taulukoista 34-35 nähdään tehtyjen ajojen tulokset. Tuloksien nähdään olevan todella hyvät. Huomataan myös, että kaikilla malleilla päästiin yhtä hyvään tulokseen.

Taulukko 34 *Painesäiliön optimoinnin tulokset (mallien vertailu).*

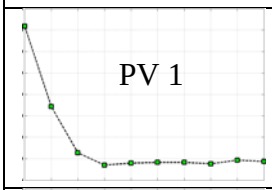
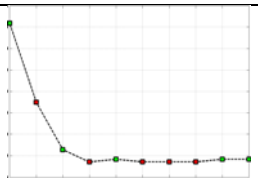
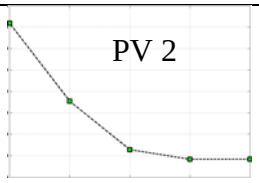
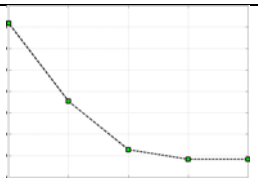
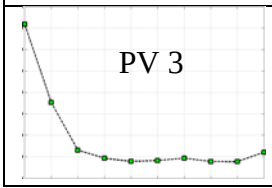
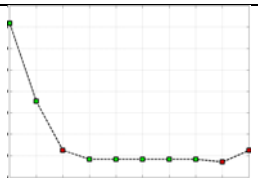
Tunniste	PV 1	PV 2	PV 3
Algoritmi	DS G/S	DS G/S	DS G/S
Malli	L	Q	PQ
Suunnittelu-muuttujat	0.9375 0.5 48.5 112	0.9375 0.5 48.5 112	0.9375 0.5 48.5 112
Tarkka arvo	6418.222	6418.222	6418.222
Mallin antama arvo	6398.491	6418.737	6465.872
ROI:n säde %	1.4817	11.25	2.2224
Residuaali	-19.730	0.515	47.651
Virhe %	0.307	0.008	0.742
Rajoiteyht. keskim. virhe	0	0	0
Käypä	kyllä	kyllä	kyllä
Käypien pis-teiden lkm	5	5	7
Iteraatio	5/10	4/5	4/10
Aika	20 s	16 s	21 s

Taulukko 35 *Painesäiliön optimoinnille kirjallisuudesta löydetyt parhaat tulokset verrattuna parhaaseen laskentatyökalulla saatuun (PV 2, TolCon=1e-2).*

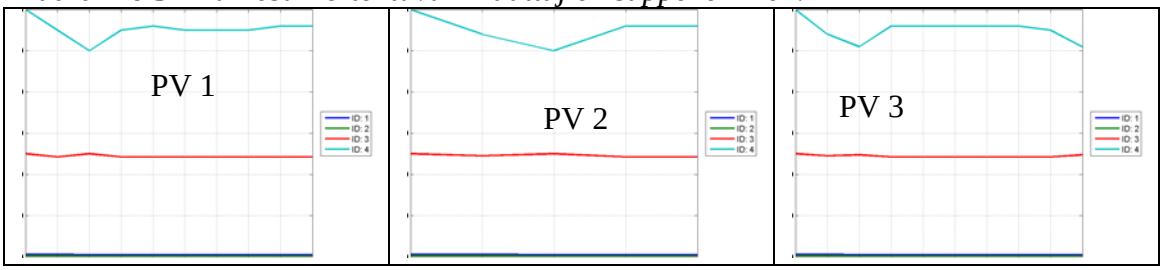
	Kirjallisuus (Loh & Papalambros 1991)	Scip	Laskentatyökalu (paras TolCon sisällä) (iter 4/5)
Rajoiteyht. keskim. virhe	0	0	0
Rajoiteyht. max virhe	0	0	0
Suunnittelu-muuttujat	1 0.5 48.5 112	0.9375 0.5 48.5 112	0.9375 0.5 48.5 112
Arvo	6788.988	**6418.222**	**6418.222**

Kohdefunktion arvo pienenee huomattavasti. Huomataan, että yhden muuttujan ollessa eri verrattuna kirjallisuudessa saatuun, näyttää vaikuttavan paljon kohdefunktioon. Lisäksi tehtävä on ratkennut nopeasti ja kohdefunktio on supennut kaikilla ajoilla hyvin. Tehokkaimmin kvadraattisilla malleilla, kuten nähdään taulukosta 36. Muuttujien suppenemisesta huomataan taulukosta 37, että x_4 muuttuu selvästi eniten.

Taulukko 36 *Painesäiliö-tehtävän kohdefunktioiden suppeneminen.*

Malli	Tarkka	Malli	Tarkka
			
			

Taulukko 37 *Painesäiliö-tehtävän muuttujien suppeneminen.*



Riippumattomille diskreeteille joukoille tehty malli toimii niin kuin pitää. Seuraavaksi kokeillaan muitakin diskreettejä malleja vielä hieman monimutkaisemmassa sekaluku-tehtävässä.

5.2.6 Ulokepalkki

Ulokepalkin tilavuus ja rajoitteet voidaan esittää (Lemonge et al. 2010)

$$\begin{aligned} V(H_i, B_i) &= 100 \sum_{i=1}^{5} H_i B_i \\ g_i(H_i, B_i) &= \sigma_i \leq 14000 N/cm^2, \qquad i = 1, \dots, 5 \\ g_{i+5}(H_i, B_i) &= H_i / B_i \leq 20, \qquad i = 1, \dots, 5 \\ g_{11}(H_i, B_i) &= \delta \leq 2.7 cm, \end{aligned} \tag{48}$$

missä H_i on ulokepalkin osan korkeus, B_i on leveys, σ_i on ulokepalkin osien jännitykset ja δ on palkin pään taipuma pystysuunnassa. Tehtävä löytyy myös MATLABin dokumentaatiosta nimellä ”Solving a Mixed Integer Engineering Design Problem Using the Genetic Algorithm”. Dokumentaatiosta nähdään myös käytetyt funktiot ja arvot. (Matlab 2013)

Taulukosta 38 nähdään tehtävien asetukset ja minkälaisia testitapauksia on muokattu alkuperäisestä tehtävästä. Alkuperäistä tehtävää on optimoitu neljässä ensimmäisessä ajossa.

Taulukko 38 *Ulokepalkin optimoinnissa käytetyt yleiset asetukset ja lähtötiedot.*

Faktoreiden lukumäärä ja tyypit	10, sekalukutehtävä, katso tyypit alta
CB 1-4	IISSSSCCCC (2 kokonaislukumuuttujaa, 4 diskreettiä ja 4 jatkuvaa muuttujaa)
CB 5	IIDDDDCCCC (2 kokonaislukumuuttujaa, 4 riippuvaa diskreettiä ja 4 jatkuvaa muuttujaa)
CB 6	IISDDSCCCC (2 kokonaislukumuuttujaa, 2 diskreettiä, 2 riippuvaa diskreettiä ja 4 jatkuvaa muuttujaa)
Rajoitusyhtälöiden lukumäärä	11
Minimoitava vaste $V(\mathbf{x})$	Tilavuus
Aloituspiste $\mathbf{x}_0$	(5, 62, 3.1, 60, 2.8, 55, 2.9, 57, 2.4, 47)
Aloituspiste käypä	kyllä
Arvot aloituspisteessä	$V(\mathbf{x}_0)$ = 92810.000000 $g_1(\mathbf{x}_0)$ = -8341.330919 $g_2(\mathbf{x}_0)$ = -7631.992868 $g_3(\mathbf{x}_0)$ = -3374.262102 $g_4(\mathbf{x}_0)$ = -3247.311828 $g_5(\mathbf{x}_0)$ = -6195.629553 $g_6(\mathbf{x}_0)$ = -1.057203 $g_7(\mathbf{x}_0)$ = -38.000000 $g_8(\mathbf{x}_0)$ = -2.000000 $g_9(\mathbf{x}_0)$ = -1.000000 $g_{10}(\mathbf{x}_0)$ = -1.000000 $g_{11}(\mathbf{x}_0)$ = -1.000000
Rajoitusyhtälöiden oikea puoli	**0**
Globaalit rajat	$x_1 \in [1, 5]$, $x_2 \in [30, 65]$, $x_3 \in [2.4, 3.1]$, $x_4 \in [45, 60]$, $x_5 \in [2.4, 3.1]$, $x_6 \in [45, 60]$, $x_7 \in [1, 5]$, $x_8 \in [30, 65]$, $x_9 \in [1, 5]$, $x_{10} \in [30, 65]$
Diskreetit joukot	$x_3 \in \{2.4, 2.6, 2.8, 3.1\}$ $x_4 \in \{45, 50, 55, 60\}$ $x_5 \in \{2.4, 2.6, 2.8, 3.1\}$ $x_6 \in \{45, 50, 55, 60\}$
Riippuvuudet (CB 5) (x_3, x_4, x_5, x_6)	(4, 3, 2, 2), (4, 4, 2, 2), (4, 3, 3, 2), (4, 4, 3, 2), (4, 3, 4, 2), (4, 4, 4, 2), (4, 3, 3, 3), (4, 3, 4, 3), (4, 4, 3, 3), (4, 3, 4, 4), (4, 4, 4, 4)
Riippuvuudet (CB 6) (x_4, x_5)	(3,2), (4,2), (3,3), (4,3), (3,4), (4,4)
Lopetusehto (konvergenssi)	1e-9 = 0.0000001 %
Maksimi iteraatiot	20
Aloitus ROI	20 %
Adaptiivinen ROI	kyllä
Suunnittelumuuttujien tarkkuus	0, 0, 1e-1, 0, 1e-1, 0, 1e-12, 1e-12, 1e-12, 1e-12
Tavoitevirhe (TolRel)	5 %
TolCon	1e-2
Techila käytössä	ei

Taulukossa 39 esitettävistä tuloksista huomataan, että kaikissa ajoissa on saatu optimiin samat diskreetit arvot ja ainoastaan jatkuvissa muuttujissa ja kokonaislukumuuttujissa on hieman eroa. Paras ratkaisu on katsottu alkuperäistä tehtävää vastaavista ajoista CB 1-4. Kahdessa viimeisessä tehtävässä on käytetty toisistaan riippuvia joukkoja, jotka on haettu etukäteen ajamalla tehtävä useampaan kertaan eri asetuksilla ja valitsemalla tuloksista käyvät riippuvat diskreetit joukot.

Taulukko 39 *Tulokset ulokepalkki-tehtäville.*

Tunniste	M	Muuttujien tyypit	**x***	*f(***x******)***	Käypä k/e (virhe)	Iter
Scip	-	IISSSSCCCC, IIDDDDCCCC, IISDDSCCCC	(3, 60, 3.1, 55, 2.6, 50, 2.281, 45.618, 1.750, 34.995)	64578	-	-
CB 1	L	IISSSSCCCC	(4, 58, 3.1, 55, 2.6, 50, 2.246, 43.952, 1.763, 35.250)	69337	k (0)	19/20
CB 2	Q	IISSSSCCCC	(3, 60, 3.1, 55, 2.6, 50, 2.263, 45.262, 1.750, 34.995)	64417	k (6e-3)	9/20
CB 3	PQ	IISSSSCCCC	(3, 60, 3.1, 55, 2.6, 50, 2.240, 43.871, 2.274, 31.073)	64942	k (5e-3)	16/20
CB 4	L/ Q	IISSSSCCCC	(3, 60, 3.1, 55, 2.6, 50, 2.251, 45.009, 1.750, 34.995)	64302	k (2e-3)	7/20
CB 5	Q	IIDDDDCCCC	(3, 60, 3.1, 55, 2.6, 50, 2.242, 44.840, 1.754, 35.088)	64259	k (2e-3)	3/20
CB 6	Q	IISDDSCCCC	(3, 60, 3.1, 55, 2.6, 50, 2.263, 45.262, 1.750, 35.000)	64417	k (6e-3)	9/20

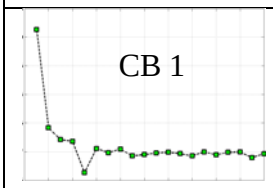
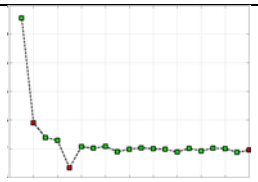
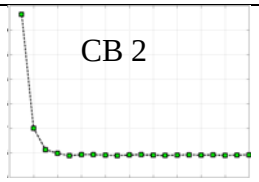
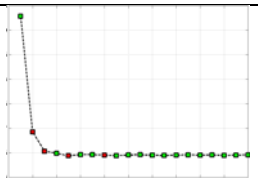
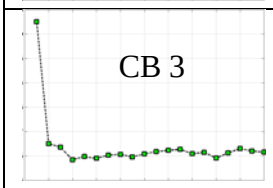
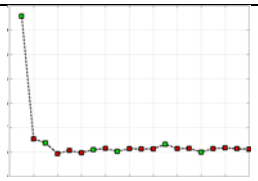
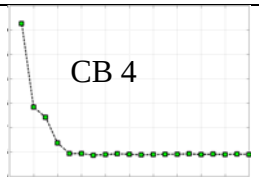
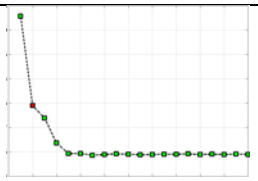
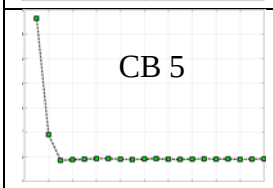
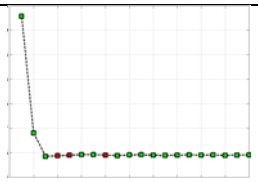
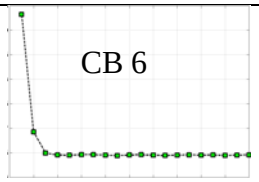
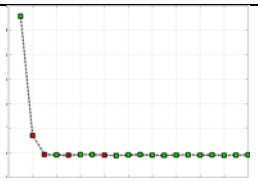
Taulukossa 40 on jälleen verrattu parasta tulosta kirjallisuudesta saatuun. Laskentatyökalun parhaassa tuloksessa rikotaan hieman enemmän rajoitusyhtälöä kuin kirjallisuudessa annetussa tuloksessa. Jos sallitaan vähän enemmän ylitystä, niin tulos paranee huomattavasti.

Taulukko 40 *Ulokepalkin optimoinnille kirjallisuudesta löydetyt parhaat tulokset verrattuna parhaaseen laskentatyökalulla saatuun (CB 4, TolCon=1e-2).*

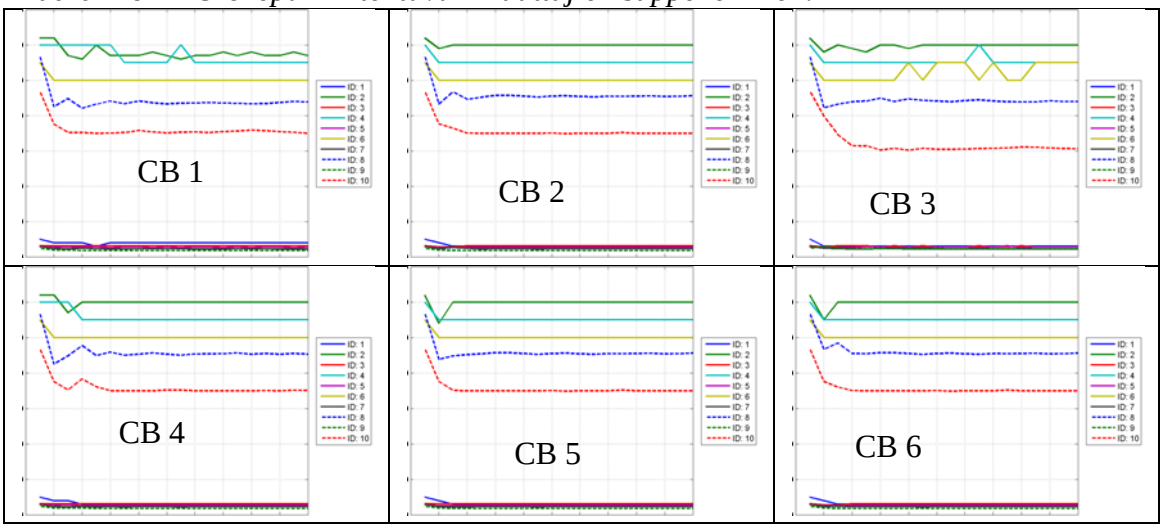
	Kirjallisuus (Thanedar & Vanderplaats 1995)	Scip	Laskentatyökalu (paras TolCon sisällä) (iter 7/20)	Laskentatyökalu (pienin virhe) (iter 15/20)
Rajoiteyht. keskim. virhe	Ei tietoa	Ei tietoa	0.001622	0 (täysin käypä)
Rajoiteyht. max virhe	0.001	Ei tietoa	0.0178	0
Suunnittelu-muuttujat	(3, 60, 3.1, 55, 2.6, 50, 2.279, 45.553, 1.750, 35.004)	(3, 60, 3.1, 55, 2.6, 50, 2.281, 45.618, 1.750, 34.995)	(3, 60, 3.1, 55, 2.6, 50, 2.251, 45.009, 1.750, 34.995)	(3, 60, 3.1, 55, 2.6, 50, 2.282, 45.637, 1.750, 34.995)
Arvo	64558	64578	**64302**	64587

Kuvaajia tarkastelemalla taulukoista 41 ja 42 nähdään, että kvadraattisella mallilla tehtävä on supennut tasaisemmin. Lineaarisella ja kvadraattisella mallilla suunnittelumuuttujat vaihtelevat selvästi enemmän kuin kvadraattisella. Ajossa CB 4 on kokeiltu muuttuvaa mallia, jossa on lähdetty liikkeelle lineaarisella mallilla ja vaihdettu tarvittaessa kvadraattiseen jos muuttujien rajoja on jouduttu nostamaan. Ratkaisu on toiminut tässä hyvin, koska on saatu yhdistettyä molempien mallien hyödyt. Lineaarista mallia on käytetty kahdessa ensimmäisessä iteraatiossa, jonka jälkeen tarkkuutta on haettu lopuilla kierroksilla kvadraattisella mallilla. Hyötynä lineaarisessa mallissa on erityisesti koepisteiden laskennan pienempi tarve eli nopeus, mutta joskus lineaarinen malli voi olla tarkempikin, tilanteissa joissa kvadraattinen malli aiheuttaa vasteeseen värähtelyä.

Taulukko 41 *Ulokepalkki-tehtävän kohdefunktioiden suppeneminen.*

Malli	Tarkka	Malli	Tarkka
			
			
			

Taulukko 42 *Ulokepalkki-tehtävän muuttujien suppeneminen.*



Edellisten diskreettien tehtävien perusteella diskreetit mallit toimivat myös implementoituna laskentatyökaluun. Diskreettejä tehtäviä, joissa käytetään laadittuja diskreettejä malleja, ei esitetä tässä enempää.

5.2.7 Kvadraattinen tehtävä

Viimeisessä testiprobleemassa optimoidaan tehtävää, jossa on vain lineaarisia ja kvadraattisia funktioita. Tässä on verrattu vain lineaarisen optimoinnin Interior point -algoritmia kvadraattisen optimoinnin algoritmiin. Lisäksi verrataan adaptiivisen ja kiinteän ROI:n säteen vaikutusta suppenemisen tehokkuuteen ja kohdefunktion tarkkuuteen.

Tehtävässä **TP113** on kvadraattiset kohde- ja rajoitefunktiot ja 10 jatkuvaa muuttujaa. (Hock & Schittkowski 1981)

$$
\begin{aligned}
\min f(\mathbf{x}) &= x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 \\
&\quad +4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 \\
&\quad +7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \\
g_1(\mathbf{x}) &= 105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0 \\
g_2(\mathbf{x}) &= -10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0 \\
g_3(\mathbf{x}) &= 8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0 \\
g_4(\mathbf{x}) &= -3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0 \\
g_5(\mathbf{x}) &= -5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0 \\
g_6(\mathbf{x}) &= -0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0 \\
g_7(\mathbf{x}) &= -x_1^2 - 2(x_2 - 2)^2 + 2x_1 x_2 - 14x_5 + 6x_6 \geq 0 \\
g_8(\mathbf{x}) &= 3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 0
\end{aligned}
\tag{49}
$$

Tehtävänannosta huomataan, että kohdefunktio on kvadraattinen ja rajoitefunktioista kolme ensimmäistä ovat lineaarisia ja loput ovat kvadraattisia. Tehtävä vaikuttaa etukä-

teen ajateltuna molemmille metamalleille helpolta, kun verrataan edellisiin erittäin epälineaarisiin tehtäviin. Taulukosta 43 nähdään tehtäviin käytetyt asetukset.

Taulukko 43 *Tehtävän TP113 optimoinnissa käytetyt yleiset asetukset ja lähtötiedot.*

Faktoreiden lukumäärä ja tyypit	10 jatkuvaa muuttujaa
Rajoitusyhtälöiden lukumäärä	8
Minimoitava vaste $f(\mathbf{x})$	Kustannus
Aloituspiste $\mathbf{x}_0$	(2, 3, 5, 5, 1, 2, 7, 3, 6, 10)
Aloituspiste käypä	kyllä
Arvot aloituspisteessä	753 $f(\mathbf{x}_0) = 753.000000$ $g_1(\mathbf{x}_0) = -76.000000$ $g_2(\mathbf{x}_0) = -117.000000$ $g_3(\mathbf{x}_0) = -12.000000$ $g_4(\mathbf{x}_0) = -105.000000$ $g_5(\mathbf{x}_0) = -5.000000$ $g_6(\mathbf{x}_0) = -9.000000$ $g_7(\mathbf{x}_0) = -4.000000$ $g_8(\mathbf{x}_0) = -10.000000$
Rajoitusyhtälöiden oikea puoli	**0**
Globaalit rajat	Rajoittamaton
Lopetusehto (konvergenssi)	1e-12
Maksimi iteraatiot	100
Aloitus ROI (säde)	10 %
Adaptiivinen ROI	kyllä / ei
Suunnittelumuuttujien tarkkuus	1e-12
Tavoitevirhe (TolRel)	5 %
TolCon	1e-4
Techila käytössä	ei

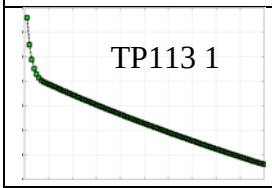
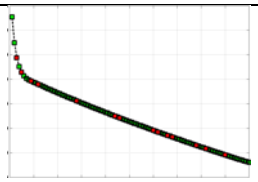
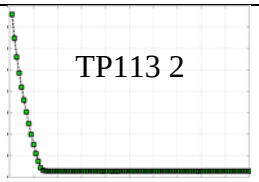
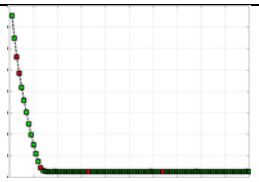
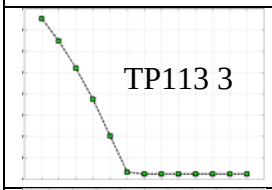
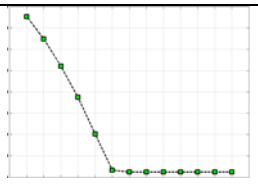
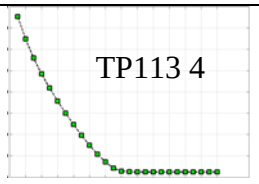
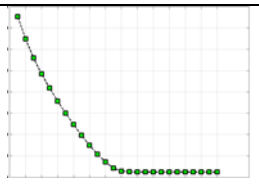
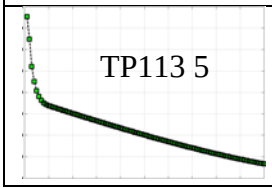
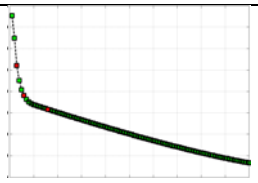
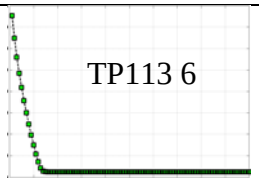
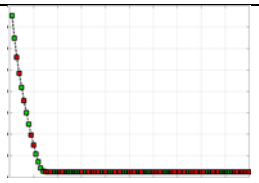
Taulukon 44 tuloksista huomataan, että ainoastaan kvadraattisella mallilla tehtävän ajo on pysähtynyt ennen asetettua maksimi-iteraatioiden määrää. Lisäksi adaptiivisella ROI:n säteellä ja lineaarisella mallilla ei ole päästy lähellekään globaalia optimia. Syy selviää katsomalla kuvaajia taulukoista 45 ja 46. Ensimmäisessä ajossa kohdefunktion suppeneminen on muutaman kierroksen jälkeen hidastunut huomattavasti. Tämä johtuu siitä, että lineaarinen malli ei ole tässä tehtävässä ollut tarpeeksi tarkka, jolloin ROI:n sädettä on pienennetty joka kierroksella. Approksimaatiovirhettä on huomattavasti, koska tähän tehtävään oli asetettu vielä suurempi tavoitevirhe kuin mitä edellisissä tehtävissä oli yleisesti käytetty.

Taulukko 44 *TP113 optimoinnin tulokset (algoritmien vertailu).*

Tunniste	TP113 1	TP113 2	TP113 3	TP113 4	TP113 5	TP113 6
Algoritmi	ML IP	ML IP	Scip	Scip	Scip	Scip
Malli	L	L	Q	Q	PQ	PQ
Adaptiivinen ROI	kyllä	ei	kyllä	ei	kyllä	ei
Suunnittelu-muuttujat	2.239988 2.596770 8.149122 5.240829 1.026816 1.411945 2.933573 7.004493 8.272785 8.318780	2.235528 2.110550 8.640160 4.924360 0.908436 1.394698 1.498287 10.000000 8.538811 8.515464	2.171996 2.363683 8.773926 5.095985 0.990655 1.430574 1.321644 9.828726 8.280091 8.375927	2.171996 2.363683 8.773926 5.095985 0.990655 1.430574 1.321644 9.828726 8.280092 8.375927	2.232273 2.575420 8.210065 5.201473 1.030153 1.428210 2.272470 8.751085 8.287916 8.366118	2.172563 2.362286 8.773689 5.095897 0.992927 1.436647 1.322739 9.829615 8.280834 8.374119
Tarkka arvo	160.043449907637	25.453375	**24.306199**	**24.306199**	66.472263	24.306530
Mallin antama arvo	160.050868	27.689636	24.306199	24.306199	66.472292	24.306125
ROI:n säde %	0.75	10	256.29	10	0.75	10
Residuaali	0.007418	2.236261	0	0	0.000029	-0.000405
Virhe %	0.004635	8.785715	0	0	0.000044	0.001668
Rajoiteyht. keskim. virhe	0.000012846856	0	0.000001262369	0.000001262532	0.000009713452	0.000001212396
Käypä	kyllä	kyllä	kyllä	kyllä	kyllä	kyllä
Käypien pis-teiden lkm	86	95	13	26	97	57
Iteraatio	100/100	24/100	10/13	17/26	100/100	41/100

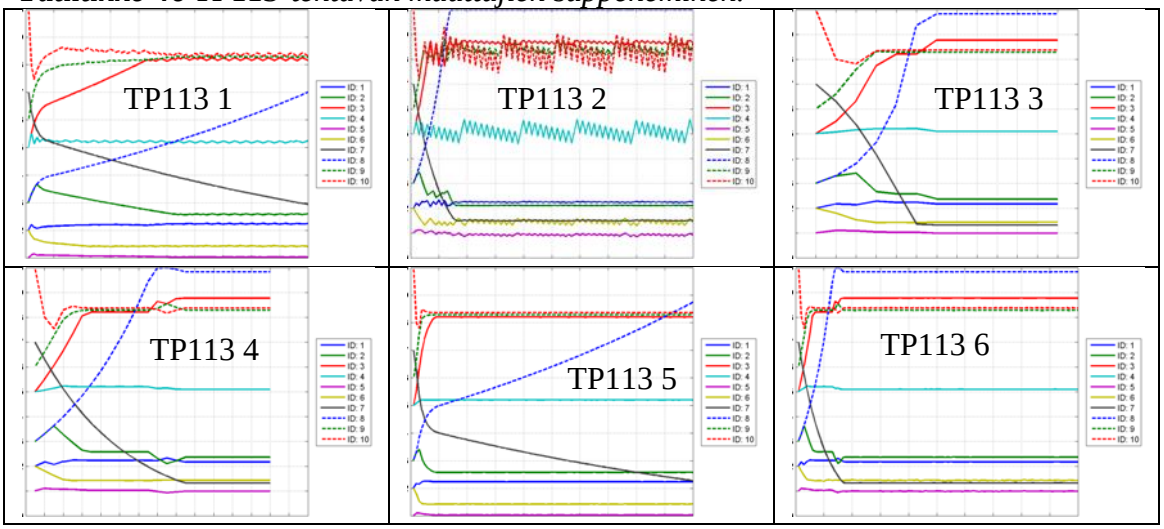
Tuloksista huomataan, että adaptiivinen ROI:n säde ei ole toiminut hyvin myöskään puhtaasti kvadraattisella mallilla. Siinä on käynyt samoin kuin lineaarisella mallilla. Tehtävän ristikkäistermit hankaloittavat vasteiden approksimointia. Kvadraattisella mallilla voidaan approksimoida kvadraattista tehtävää täysin, mikä on itsestään selvää. Huomataan, että kvadraattisella mallilla on tässä kasvatettu ROI:n sädettä joka kierroksella maksimin verran, jolloin ROI:n säde on lopussa peräti 256 %. Kiinteällä 10 % ROI:n säteellä on päästy taas kaikilla malleilla hyvään tulokseen. Lineaarisen mallin huonomman tarkkuuden takia on vain saatu hieman heikompi arvo. Kiinteällä ROI:n säteellä lineaarinen ja puhtaasti kvadraattinen suppenevat myös aluksi melko tehokkaasti, jonka jälkeen epätarkkuuden takia jäädään samalle tasolle.

Taulukko 45 *TP113-tehtävän kohdefunktioiden suppeneminen.*

Malli	Tarkka	Malli	Tarkka
			
			
			

Muuttujien kuvaajista taulukosta 46 huomataan hyvin lineaarisen mallin epätarkkuus, sillä muuttujat värähtelevät voimakkaasti, erityisesti kiinteällä aliavaruuden koolla. Kvadraattisella mallilla ja adaptiivisella aliavaruuden koolla muuttujat suppenevat tehokkaasti, mikä on selvää tässä tehtävässä.

Taulukko 46 *TP113-tehtävän muuttujien suppeneminen.*



Taulukosta 47 nähdään, että kvadraattisella mallilla päästään jopa hieman parempaan tulokseen kuin kirjallisuudessa on päästy jos hyväksytään suurempi rajoitusyhtälöiden rikkominen. Pieni rajoitusyhtälöiden rikkominen kvadraattisella mallilla, johtunee Scip-algoritmin ja MATLABin laskentatarkkuuksista. Taulukon 47 vertailussa huomataan, että laskentatyökalulla on saatu täsmälleen sama ratkaisu kuin SCIP-algoritmilla suoraan ajettuna.

Taulukko 47 *TP113 optimoinnille kirjallisuudesta löydetyt parhaat tulokset verrattuna parhaaseen laskentatyökalulla saatuun (TolCon=1e-4).*

	Kirjallisuus (Hock & Schittkowski 1981)	Scip	Laskentatyökalu (paras TolCon sisällä) TP113 3 (iter 10/13)	Laskentatyökalu (pienin virhe) TP113 2 (iter 24/100)
Rajoiteyht. keskim. virhe	Ei tietoa	1e-6	1e-6	0
Rajoiteyht. max virhe	12e-9	5e-6	5e-6	0
Suunnittelu-muuttujat	2.171996 2.363683 8.773926 5.095984 0.990655 1.430574 1.321644 9.828726 8.280092 8.375927	2.171996 2.363683 8.773926 5.095985 0.990655 1.430574 1.321644 9.828726 8.280091 8.375927	2.171996 2.363683 8.773926 5.095985 0.990655 1.430574 1.321644 9.828726 8.280091 8.375927	2.235528 2.110550 8.640160 4.924360 0.908436 1.394698 1.498287 10.000000 8.538811 8.515464
Arvo	24.3062091	24.3061990	**24.3061990**	25.4533754

Muun muassa tätä tehtävää koetettiin optimoida myös Gurobilla käyttämällä kvadraattista metamallia. Tämän ajon tuloksia ei tässä esitellä tarkemmin. Kahdella ensimmäisellä iteraatiolla tehtävä etenee hyvin, jonka jälkeen Gurobi palauttaa virheen, että jokin kvadraattisista funktioista ei ole positiivisesti semidefiniitti. Tämän jälkeen laskentatyökalu vaihtaa automaattisesti lineaariseen malliin ja optimoi uudestaan MATLAB Interior point -algoritmilla, jottei tehtävän suoritus katkea. Laskentatyökalu vaihtaa kuitenkin tämän jälkeen takaisin alkuperäiseen malliin ja optimointialgoritmiin. Tämä toistuu iteraatiokierroksien loppuun asti. Laskentatyökalun ”model_history.txt”-raportista nähdään mitä mallia on käytetty eri iteraatiokierroksilla. Tehtävän optimointi meni kuitenkin paremmin kuin käyttämällä lineaarista mallia, sillä kvadraattisella mallilla saatiin tarkasti approksimoitua vastetta kierroksilla, jolloin tehtävä oli konveksi. Eli kuten luvussa 4.2 todettiin, niin Gurobilla ei voida ratkaista epäkonvekseja tehtäviä. Vaikka työkalu osaa vaihtaa lineaariseen malliin epäkonveksissa tapauksessa, niin silti ei kannata käyttää Gurobia kvadraattiseen optimointiin, koska lineaarista mallia varten pitää laskea koepisteet uudestaan.

5.3 FEM-mallin vasteiden optimointi

Hyvien testiprobleemien tulosten jälkeen testattiin laskentatyökalua vielä sen oikeaan käyttötarkoitukseen eli FEM-rakennemallin optimointiin. Olennaisena erona FEM-mallin lisäksi on se, että nyt ei ole käytettävissä tulosta, johon verrata. Tavoitteena on siis vain parantaa lähtötilannetta mahdollisimman paljon.

FEM-mallien avulla haluttiin myös kokeilla erilaisia laskentatyökalun asetuksia, jotta nähdään, että pitävätkö testiprobleemien kanssa tehdyt päätelmät paikkaansa myös todellisten tehtävien kanssa. Seuraavaksi esitellään tuloksia, mitä on saatu optimoimalla kahta FEM-mallia, joista ensimmäinen on staattinen lastiluukku ja toinen dynaaminen palkki, jota kuormitetaan ajan suhteen. Ensimmäistä FEM-mallia optimoidaan samoilla asetuksilla kuin testiprobleemien ensimmäistä tehtävää, joka oli kierrejousen tilavuuden minimointi. Toista FEM-mallia ei voitu testata yhtä laajasti, koska sitä ei saatu toimimaan Techila-laskentaympäristön kanssa. Dynaaminen palkki on tehty siten, että siinä on käytetty ANSYS- ja SolidWorks-suunnitteluohjelmia, kun taas lastiluukkutehtävässä on käytetty pelkästään ANSYS-ohjelmaa. ANSYS-malli, jossa on SolidWorks-kytkentä, jouduttiin ajamaan paikallisesti ilman Techila-laskentaympäristöä, jolloin tehtävien ajaminen kesti todella pitkään.

5.3.1 Lastiluukku

Lastiluukun optimointitehtävässä on 17 faktoria ja 37 vastetta, joista massaa minimoidaan. Muut vasteet toimivat tehtävän rajoitusyhtälöinä. Seuraavaan taulukkoon 48 on koottu tehtävän lähtötiedot ja optimoinnissa käytetyt asetukset.

Käytetyt asetukset ja lähtötiedot nähdään taulukoista 48-49. Taulukossa 49 on esitetty myös ajojen tulokset. Tuloksia on analysoitu lyhyesti liitteessä 15 vastaavasti kuin kierrejousen optimoinnissa. Liitteissä 16-17 esitetään jokaisen ajon vasteiden ja suunnittelumuuttujien kuvaajat. Lastiluukusta esitellään vain todellisten vasteiden vertailu, koska mallin antama vaste on täysin vastaava melkein kaikissa ajoissa. Rajoitefunktioiden kuvaajia ei niiden suuren määrän takia tässä esitetä. Tässä tehtävässä ei esitetä kuvaa optimointikohteesta, koska kyseessä on erään yrityksen oikea tuote. Tästä syystä ei näytetä myöskään muita kuvakaappauksia ANSYS-ohjelmasta. Seuraavassa dynaamisen palkin optimoinnissa esitetään kuvia myös ANSYS-ohjelmasta.

Ensimmäiset 27 tehtävää on ajettu jatkuvilla suunnittelumuuttujilla ja loput neljä on ajettu kokonaislukutehtävänä. Iteraatioiden maksimimäärä rajoitettiin 30 ja tehtävässä käytettiin Techila-laskentaympäristöä vasteiden laskemiseen. Suunnittelumuuttujien tarkkuus rajattiin tarkkuuteen 1e-6, koska käytännössä yleensä ei tarvita tarkempaa.

Taulukko 48 *Lastiluukun optimoinnissa käytetyt yleiset asetukset ja lähtötiedot.*

Faktoreiden lukumäärä ja tyypit	17 jatkuvaa (Bufo 1-27) tai 17 kokonaislukumuuttujaa (Bufo 28-31)
Rajoitusyhtälöiden lukumäärä	36
Minimoitava vaste $f(\mathbf{x})$	Lastiluukun kokonaismassa
Aloituspiste	(9, 8, 10, 9, 10, 10, 11, 8, 9, 10, 10, 7, 9, 8, 9, 12, 8)
Aloituspiste käypä	kyllä
Vasteet aloituspisteessä	$f(\mathbf{x}_0) = 22208.563626$ $g_1(\mathbf{x}_0) \ldots g_{36}(\mathbf{x}_0) < 1$
Rajoitusyhtälöiden oikea puoli	**1**
Globaalit rajat	$x_1 \ldots x_8 \in [8, 12]$ $x_9 \ldots x_{12} \in [7, 12]$ $x_{13} \ldots x_{17} \in [8, 12]$
Lopetusehto (konvergenssi)	1e-3 = 0.1 %
Maksimi iteraatiot	30
Aloitus ROI	10 %
Adaptiivinen ROI	kyllä
Suunnittelumuuttujien tarkkuus	1e-6
Tavoitevirhe (TolRel)	1 %
TolCon	1e-3
Techila käytössä	kyllä

Taulukko 49 *Lastiluukun optimointiajot.*

Tunniste	M	Algoritmi	Kuvaus	m / kg	Käypä k/e (virhe)	Aika / min	Iter
Bufo 1	L	ML IP	Katso taulukko 48	20112	k (0)	40	8/8
Bufo 2	L	MLSX	Katso taulukko 48	20112	k (0)	41	8/8
Bufo 3	L	ML AS	Katso taulukko 48	20112	k (0)	41	8/8
Bufo 4	L	Gurobi	Katso taulukko 48	20112	k (0)	44	8/8
Bufo 5	Q	Scip	Katso taulukko 48	20098	k (2e-4)	80	5/6
Bufo 6	PQ	Scip	Katso taulukko 48	20094	k (1e-4)	46	8/8
Bufo 7	L	ML IP	Bufo 1, kiinteä ROI	20190	k (0)	22	6/6
Bufo 8	L	ML IP	Bufo 1, kiinteä ROI 20 %	20394	k (0)	35	3/8
Bufo 9	L	ML IP	Bufo 1, kiinteä ROI 50 %	20432	k (0)	15	4/4
Bufo 10	L	ML IP	Bufo 1, TolRel 5 %	20234	k (2e-5)	19	5/5
Bufo 11	L	ML IP	Bufo 1, TolRel 10 %	20207	k (0)	67	10/16
Bufo 12	Q	Scip	Bufo 5, kiinteä ROI	20098	k (1e-4)	69	5/6
Bufo 13	Q	Scip	Bufo 5, kiinteä ROI 20 %	22209	k (0)	52	1/4
Bufo 14	Q	Scip	Bufo 5, kiinteä ROI 50 %	22209	k (0)	37	1/3
Bufo 15	Q	Scip	Bufo 5, TolRel 5 %	20100	k (2e-4)	137	5/5
Bufo 16	Q	Scip	Bufo 5, TolRel 10 %	20938	k (3e-4)	93	2/5
Bufo 17	PQ	Scip	Bufo 6, kiinteä ROI	**20091**	k (6e-4)	36	6/6
Bufo 18	PQ	Scip	Bufo 6, kiinteä ROI 20 %	20449	k (2e-4)	33	2/7
Bufo 19	PQ	Scip	Bufo 6, kiinteä ROI 50 %	20282	k (4e-4)	58	2/11
Bufo 20	PQ	Scip	Bufo 6, TolRel 5 %	20099	k (7e-4)	28	4/5

Bufo 21	PQ	Scip	Bufo 6, TolRel 10 %	20930	k (6e-4)	42	2/7
Bufo 22	L	ML IP	Bufo 1, aloituspiste epäkäypä $\mathbf{x} = 9$	20122	k (0)	58	15/15
Bufo 23	Q	Scip	Bufo 5, aloituspiste epäkäypä $\mathbf{x} = 9$	20110	k (2e-4)	123	8/9
Bufo 24	PQ	Scip	Bufo 6, aloituspiste epäkäypä $\mathbf{x} = 9$	20094	k (2e-4)	90	12/12
Bufo 25	L	ML IP	Bufo 1,TolRel 0.1 %	20096	k (0)	48	13/13
Bufo 26	Q	Scip	Bufo 5,TolRel 0.1 %	20095	k (2e-5)	89	6/7
Bufo 27	PQ	Scip	Bufo 6,TolRel 0.1 %	20094	k (2e-5)	74	12/12
Bufo 28	L	ML IP + DH	Bufo 1, $\mathbf{x} \in \mathbb{Z}$	20521	k (5e-5)	44	9/9
Bufo 29	L	Gurobi	Bufo 4, $\mathbf{x} \in \mathbb{Z}$	20521	k (5e-5)	46	10/13
Bufo 30	Q	Scip	Bufo 5, $\mathbf{x} \in \mathbb{Z}$	20602	k (0)	61	4/5
Bufo 31	PQ	Scip	Bufo 6, $\mathbf{x} \in \mathbb{Z}$	20521	k (5e-5)	27	4/5

Taulukon 49 ajolistauksesta huomataan, että kaikki löydetyt ratkaisut ovat virhetoleranssin sisällä ja monessa ajossa on löydetty täysin käypä optimiratkaisu. Lisäksi huomataan, että kaikissa ajoissa on lopetettu optimointi ennen asetettua iteraatioiden maksimimäärää. Konvergenssitoleranssi on siten ollut sopivan pieni. Techilan ansiosta tehtäviin ei ole kulunut paljoa aikaa. Kvadraattisiin tehtäviin on koepisteiden määrän ollessa suuri kulunut selvästi paljon enemmän aikaa, mutta pysynyt silti kohtuullisena. Liitteen 16 kuvaajista nähdään, että taulukon 48 lähtötiedoilla ajetuissa ajoissa (Bufo 1-6) kohdefunktio suppeni kaikilla hyvin, mutta kvadraattisella mallilla selvästi tehokkaimmin. Adaptiivinen ROI:n säde näyttäisi toimivan tässä hyvin, vaikka ROI:n sädettä pienennetään joka kierroksella kvadraattista mallia lukuun ottamatta, jolla sitä vähän myös välillä kasvatetaan.

Tarkastellaan kiinteän ROI:n säteellä saatuja tuloksia, jotta nähdään paremmin approksimoidun mallin tarkkuus. Liitteen kuvaajista nähdään, että kiinteällä 10 % ROI:n (Bufo 7) säteellä optimointi lineaarisella mallilla on tehokkaampi. Tuloksia katsomalla huomataan kuitenkin, että tulos heikkenee hieman. 20 % ROI:n (Bufo 8) säteellä suppenee ensin hyvin, mutta sitten neljännellä kierroksella kohdefunktion arvo hyppää huonommaksi, josta jatketaan monta kierrosta huonompaan suuntaan, kunnes lopuksi taittuu laskuun. Tällä ROI:n säteellä malli ei approksimoi hyvin vaikka pisteet ovat virhetoleranssin sisällä. Optimointialgoritmi ei löydä siten mallista muodostetusta optimointitehtävästä välttämättä aina parempaa kuin mikä oli edellinen tulos. Tulos heikkeni selvästi 10 % säteellä saatuun verrattuna. Lineaarisella mallilla kiinteän ROI:n säteen ollessa peräti 50 % (Bufo 9), päästiin muutamalla iteraatiolla lähelle optimia, mutta tulos heikkeni liikaa. Mielenkiintoista tässä oli se, että kaikkein suurimmalla ROI:n säteellä ei tullut värähtelyä vasteeseen. Suunnittelumuuttujien kuvaajasta liitteestä 17 huomataan hyvin, kuinka tehokkaasti optimaalisia suunnittelumuuttujia haetaan.

Kvadraattisella mallilla ja 10 % kiinteällä ROI:n säteellä (Bufo 12) saatiin melkein yhtä hyvä tulos kuin adaptiivisella. Suuremmilla kiinteillä arvoilla (Bufo 13-14) kohde-

funktio suppenee tehokkaasti, mutta lasketut vasteet eivät ole asetetun virhetoleranssin sisällä. Parhaat tulokset rikkovat virhetoleranssia vain hyvin vähän, joten suunnittelija voisi halutessaan hyväksyä myös ne.

Puhtaasti kvadraattisella mallilla saadaan 10 % kiinteällä ROI:n säteellä (Bufo 12) ajojen paras tulos 20091 kg kuudella iteraatiokierroksella ja aikaa meni vain 36 minuuttia. Suuremmat ROI:n säteet (Bufo 17-19) eivät tuota hyvää tulosta. Puhtaasti kvadraattisella mallilla vasteisiin tulee värähtelyä kun ROI:n säde on 20 tai 50 % ja tulokset ovat epätarkempia. Kohdefunktio suppenee kuitenkin aluksi nopeasti kunnes jää huonommalla tasolle kuin 10 % ROI:n säteellä.

Kaikilla malleilla löydetään suuremmallakin ROI:n säteellä käypä joukko optimoitavaksi joka iteraatiolla, eikä jouduta siten vaihtamaan Interior point -algoritmiin niin kuin kävi kierrejousen kanssa.

Optimoitavan lastiluukun vasteet eivät edellisten tulosten perusteella vaikuta kovinkaan epälineaarisilta. Puhtaasti kvadraattisella saatu hyvä tulos viittaisi siihen, että muuttujilla ei olisi ainakaan paljoa ristikkäisvaikutuksia.

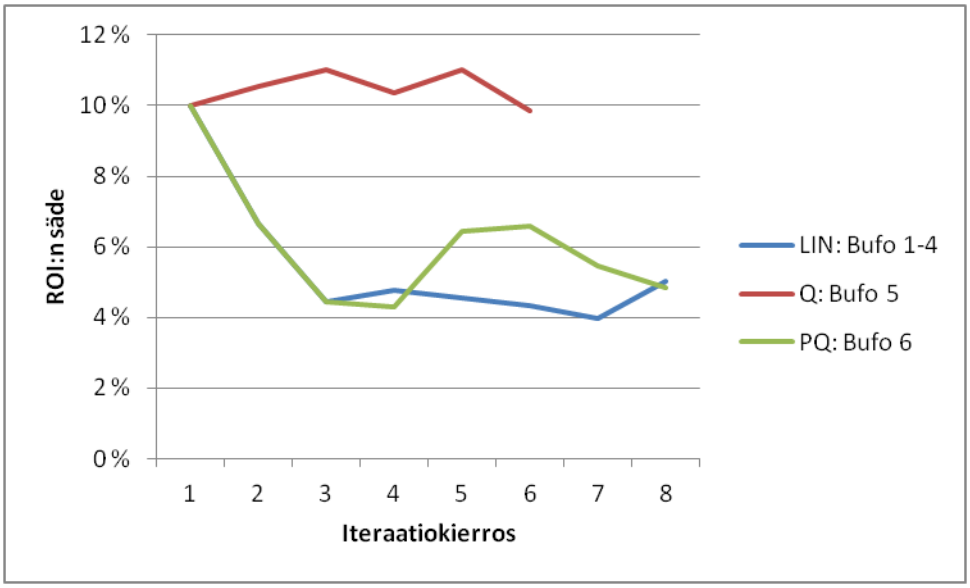
Seuraavaksi haetaan optimointiin tehokkuutta nostamalla tavoitevirhettä. Tuloksista huomataan, että 5 % tavoitevirhe (Bufo 10) lineaarisella mallilla antaa vastaavan tuloksen kuin saatiin 10 % kiinteällä ROI:n säteellä. Suurempi 10 % tavoitevirhe (Bufo 11) on jo liikaa, koska siitä aiheutuu värähtelyä, jolloin myös tarvittavien iteraatioiden määrä kasvaa huomattavasti. Tulos on kuitenkin melkein yhtä hyvä kuin kiinteällä 10 % säteellä. Lastiluukkutehtävässä kokeiltiin myös hyvin pientä 0.1 % tavoitevirhettä. Tuloksista nähdään, että sillä suppenee selvästi hitaammin kuin lähtötiedoissa annetulla 1 % tavoitevirheellä. Tulos kuitenkin parani hieman siihen verrattuna.

Kvadraattisella mallilla 5 % tavoitevirhe nopeuttaa hieman suppenemista, mutta virhetoleranssia rikotaan kahdella iteraatiolla. Viimeinen iteraatio on kuitenkin virhetoleranssin sisällä ja tulos on melkein sama, kuin 1 % prosentin tavoitevirheellä. Tavoitevirheen nostamisella 10 % ei ole juuri vaikutusta suppenemiseen, mutta mallin approksimaatio on heikompi, sillä vain ensimmäisellä iteraatiolla saadaan virhetoleranssin sisällä oleva optimi. Virhetoleranssin sisään jäänyt ratkaisu on todella huono. Taas jos hyväksytään suurempi rajoitusyhtälöiden rikkominen, niin viimeisen kierroksen optimi on hyvä tulos. Hyvin pienellä tavoitevirheellä käy samalla lailla kuin lineaarisella mallilla eli tulos tarkentuu, mutta vaatii tässä vain yhden iteraation enemmän kuin 1 % tavoitevirheellä.

Puhtaasti kvadraattisella mallilla suppenee myös tehokkaammin suuremmalla tavoitevirheellä, mutta tästä aiheutuu epätarkkuutta. Värähtelyä vasteisiin ei kuitenkaan ilmaannu niin kuin kävi lineaarisella mallilla, kun tavoitevirhe oli 10 %. Pienimmällä testatulla tavoitevirheellä saadaan tarkka tulos, mutta iteraatioiden määrän huomattavalla lisäyksellä.

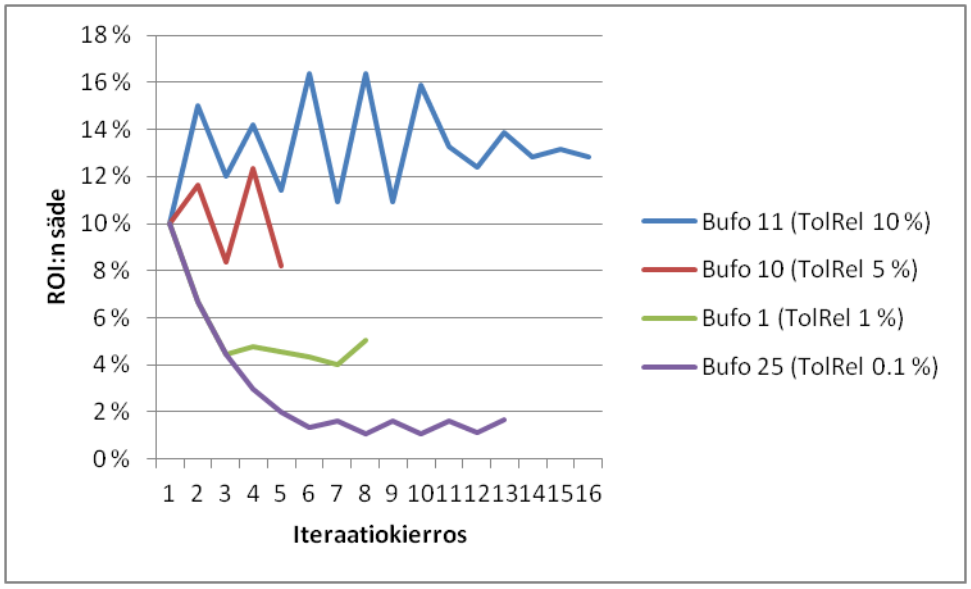
Sopivana tavoitevirheenä voidaan tulosten perusteella pitää kaikille tehtäville 1 %, mutta jos sallitaan suurempi rajoitusyhtälöiden ylitys, niin voidaan käyttää hieman suurempaakin. Tällöin tehtävä myös suppenee nopeammin.

Kuvista 26-29 nähdään, miten ROI:n säde muuttuu lastiluukkutehtävissä iteraatiokierrosten aikana eri tavoitevirheellä. Kierrejousen optimoinnissa ei nähty selkeää eroa tavoitevirheen nostamisella. Lastiluukkutehtävässä sillä on selvä vaikutus ROI:n säteeseen, kuten edellä todettiin. ROI:n säteen nostaminen nopeuttaa yleensä suppenemista. Kuvassa 26 vertaillaan ensin malleja keskenään hyväksi todetulla 1 % tavoitevirheellä. Kuvasta 26 nähdään, että lineaaristen tehtävien (Bufo 1-4) kuvaajat ovat samat. ROI:n säde pienenee kaikissa tapauksissa heti noin 4 %, jonka jälkeen se pysyy lineaarisissa tehtävissä melkein samana iteraatioiden loppuun asti. Kvadraattisessa optimoinnissa (Bufo 5) ROI:n säde pysyy melkein samana iteraatioiden loppuun asti. Puhtaasti kvadraattisen mallin optimoinnissa (Bufo 6) ROI:n säde tippuu samalle tasolle kuin lineaarisella mallilla, jonka jälkeen se vaihtelee 4 ja 6 % välillä.



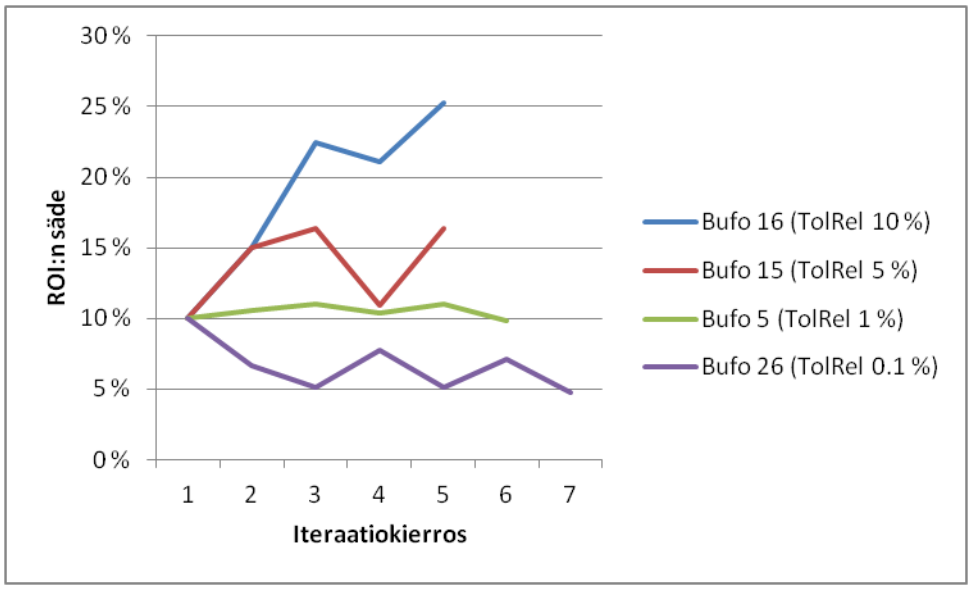
Kuva 26 *Lastiluukkutehtävän ROI:n säteen muuttuminen iteraatiokierrosten aikana kun tavoitevirhe TolRel=1 %.*

Kuvasta 27 nähdään hyvin tavoitevirheen nostamisen vaikutus lineaariseen malliin. Hyvin suuri tavoitevirhe aiheuttaa lineaarisella mallilla vasteisiin värähtelyä, kuten edellä todettiin. Tämä näkyy myös ROI:n säteen värähtelynä.



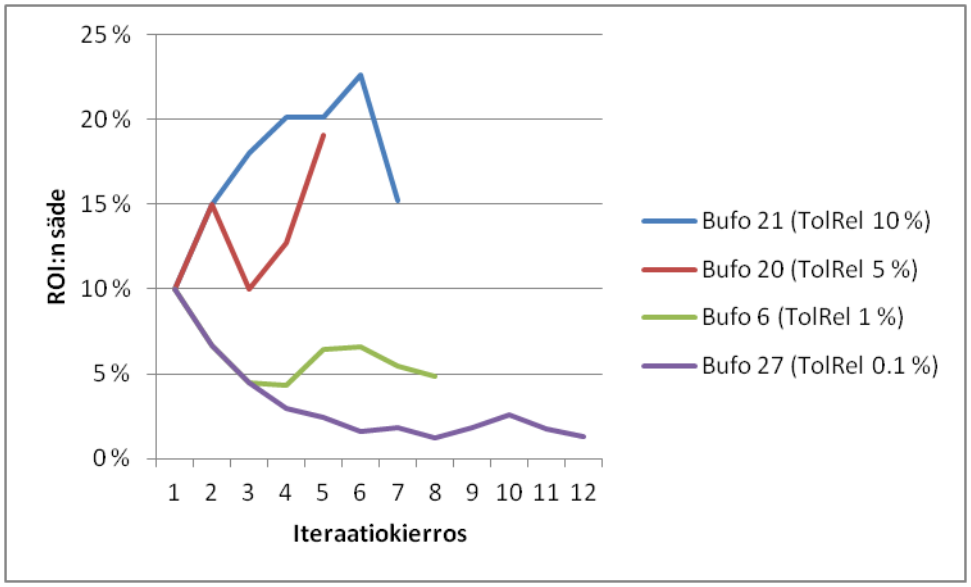
Kuva 27 *Tavoitevirheen vaikutus lastiluukkutehtävän ROI:n säteeseen iteraatiokierrosten aikana lineaarisella mallilla.*

Kvadraattisella mallilla tavoitevirheen nostamisella on maltillisempi vaikutus ROI:n säteeseen kuin lineaarisella mallilla. Tämä nähdään hyvin myös kuvasta 28. Kuvasta huomataan, että 10 % tavoitevirheellä ROI:n sädettä nostetaan melkein joka kierroksella ja lopuksi se on peräti 25 %, mutta sitä ei enää käytetä, koska iterointi pysähtyy asetettuun konvergenssitoleranssiin. Suurella tavoitevirheellä rikotaan tässä vain hieman asetettua virhetoleranssia.



Kuva 28 *Tavoitevirheen vaikutus lastiluukkutehtävän ROI:n säteeseen iteraatiokierrosten aikana kvadraattisella mallilla.*

Puhtaasti kvadraattisella mallilla tavoitevirheen vaikutus ROI:n säteeseen on vastaava kuin kvadraattisella mallilla. Vastaavasti siitä aiheutuu myös hieman virhetoleranssin ylitystä.



Kuva 29 *Tavoitevirheen vaikutus lastiluukkutehtävän ROI:n säteeseen iteraatiokierrosten aikana puhtaasti kvadraattisella mallilla.*

Ajoissa Bufo 22-24 on lähdetty liikkeelle epäkäyvästä pisteestä käyttämällä adaptiivista ROI:n sädettä. Kaikilla päästiin todella hyvään tulokseen. Epäkäyvästä aloituksesta päästiin käypään ratkaisuun käyttämällä Interior point -algotrimia tarvittaessa. Käypä alue löydettiin kaikilla malleilla jo toisella kierroksella eli Interior point -algoritmia käytettiin kvadraattisilla malleilla vain kerran.

Seuraavaksi on analysoitu tarkemmin algoritmin vaikutusta optimoinnin tarkkuuteen ja tehokkuuteen. Taulukosta 50 nähdään kuuden ensimmäisen ajon tulokset, joissa käytettiin lähtöasetuksia eri malleille ja algoritmeille.

Taulukko 50 *Lastiluukun optimoinnin tulokset (algoritmien vertailu).*

Tunniste	Bufo 1	Bufo 2	Bufo 3	Bufo 4	Bufo 5	Bufo 6
Algoritmi	ML IP	ML SX	ML AS	Gurobi	Scip	Scip
Malli	L	L	L	L	Q	PQ
Suunnittelu-muuttujat	8.3789 8.0000 8.7192 8.4489 9.0584 9.8894 10.5353 8.0198 7.0000 7.0000 9.5429 7.0000 8.0000 8.0000 8.9415 11.7594 8.0000	8.3789 8.0000 8.7192 8.4489 9.0584 9.8894 10.5353 8.0198 7.0000 7.0000 9.5429 7.0000 8.0000 8.0000 8.9415 11.7594 8.0000	8.3789 8.0000 8.7192 8.4489 9.0584 9.8894 10.5353 8.0198 7.0000 7.0000 9.5429 7.0000 8.0000 8.0000 8.9415 11.7594 8.0000	8.3789 8.0000 8.7192 8.4489 9.0584 9.8894 10.5353 8.0198 7.0000 7.0000 9.5429 7.0000 8.0000 8.0000 8.9415 11.7594 8.0000	8.3654 8.0000 8.7100 8.4399 9.0394 9.8801 10.5205 8.0176 7.0000 7.0000 9.5274 7.0000 8.0000 8.0000 8.9218 11.6926 8.0000	8.3585 8.0000 8.7041 8.4335 9.0401 9.8661 10.5209 8.0165 7.0000 7.0000 9.5197 7.0000 8.0000 8.0000 8.9212 11.7156 8.0000
Tarkka arvo	20112.02	20112.02	20112.02	20112.02	20098.43	**20094.05**
Mallin antama arvo	20112.02	20112.02	20112.02	20112.02	20098.43	20094.05
ROI:n säde %	3.99	3.99	3.99	3.99	10.36	5.47
Residuaali	7e-6	7e-6	7e-6	7e-6	-3e-5	2e-5
Virhe %	0	0	0	0	0	0
Rajoiteyht. keskim. virhe	0	0	0	0	0.000155	0.000149
Rajoiteyht. max virhe	0	0	0	0	0.005389	0.002626
Käypä	kyllä	kyllä	kyllä	kyllä	kyllä	kyllä
Käypien pis-teiden lkm	8	8	8	8	6	8
Iteraatio	8 / 8	8 / 8	8 / 8	8 / 8	5/6	8/8
Aika	40 min, 3 s	41 min, 29 s	40 min, 48 s	43 min, 48 s	1 h, 20 min	46 min

Tässä FEM-rakennemallin optimoinnissa huomataan sama kuin testiprobleemien kanssa, että lineaarisilla algoritmeilla tulee samat tulokset. Lastiluukkutehtävässä myös Interior point -algoritmi antaa saman tuloksen kuin muut. Laskenta-ajoissakaan ei havaita isompaa eroa. Itse optimointialgoritmin tekemä optimointi menee alle sekunnissa kaikilla algoritmeilla näin pienessä tehtävässä. Tehtävissä aikaa vie eniten tietysti vasteiden laskenta ja seuraavaksi eniten vie aikaa suunnittelumatriisien laskeminen, jos koepisteitä

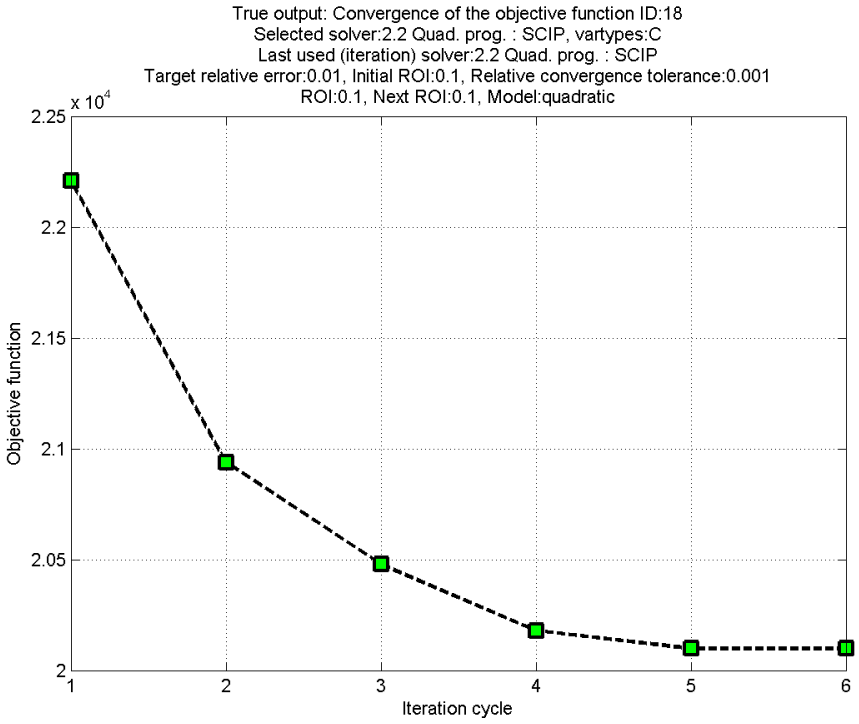
on paljon. Laskentatyökalun piirtämiin kuvaajiin menee vain muutama sekunti joka iteraatiokierroksella.

Algoritmien vertailussa huomataan myös, että kvadraattisella mallilla on vaatinut vähiten iteraatiokierroksia, mutta suuremman koepistemäärän takia on kestänyt noin kolme kertaa pidempään kuin lineaarisella ja puhtaasti kvadraattisella mallilla. Puhtaasti kvadraattisella mallilla saatiin tässä paras tulos ja aikaa meni melkein yhtä vähän kuin lineaarisella mallilla.

Lineaarinen malli vaatii tässä tehtävässä 18 koepistettä, joka on semmoinen määrä, jonka Techila-laskentaympäristöllä saa usein kerralla laskettua. Puhtaasti kvadraattisella mallilla koepisteitä tarvitaan lastiluukun optimoinnissa 35 ja kvadraattisella peräti 171.

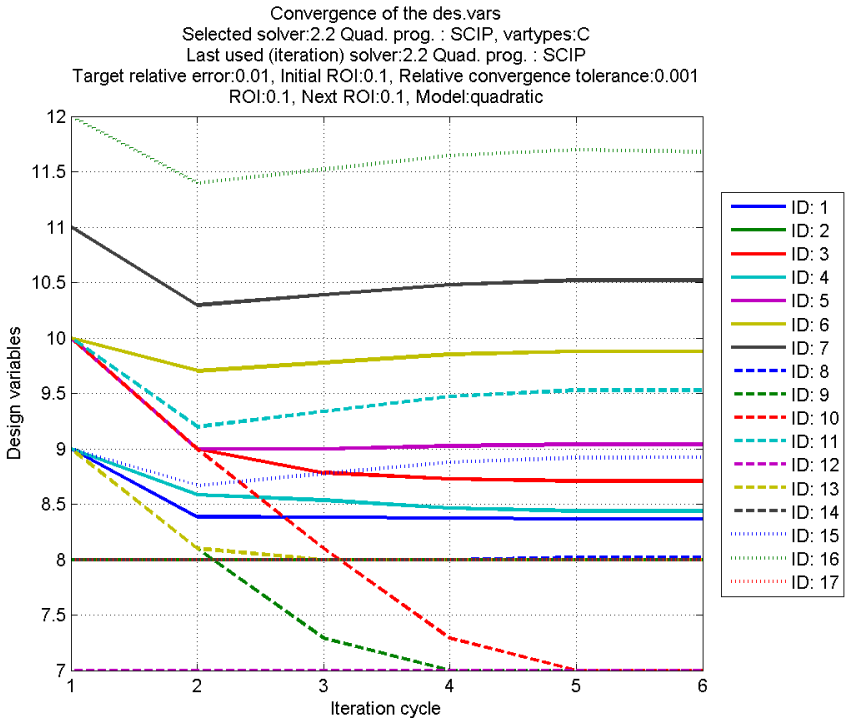
Taulukosta 50 huomataan myös, että malli approksimoi hyvin vasteita käytetyllä ROI:n säteellä. Kvadraattisessa tehtävässä 10 % ROI:n säteelläkin on saatu tarkka tulos.

Kuvassa 30 näytetään parhaimman tuloksen kuvaaja kohdefunktion suppenemisesta. Huomataan, että puhtaasti kvadraattisella mallilla suppeneminen on ollut tehokasta ja malli on approksimoidut tarkasti vasteita, sillä kaikki iteraation pisteet ovat virhetoleranssin sisällä.



Kuva 30 *Bufo 12, kohdefunktion suppeneminen (todellinen vaste).*

Kuvasta 31 nähdään miten suunnittelumuuttujat ovat muuttuneet iteraatiokierrosten aikana. Huomataan, että eniten on muuttunut suunnittelumuuttuja, jonka ID on 10. Lisäksi havaitaan, että ainakin kahden muuttujan arvo pysyy iteraatioiden aikana samana.



Kuva 31 *Bufo 12, muuttujien suppeneminen.*

Lastiluukusta ajettiin lopuksi vielä neljä ajoa kokonaislukutehtävänä. Mielenkiinnon kohteena on nähdä, miten onnistuu kokonaislukutehtävät FEM-rakennemallin kanssa sekä kuinka tarkkaan tulokseen päästään hakemalla ensin jatkuva optimi ja siitä binäärioptimoinnilla kokonaislukuratkaisu.

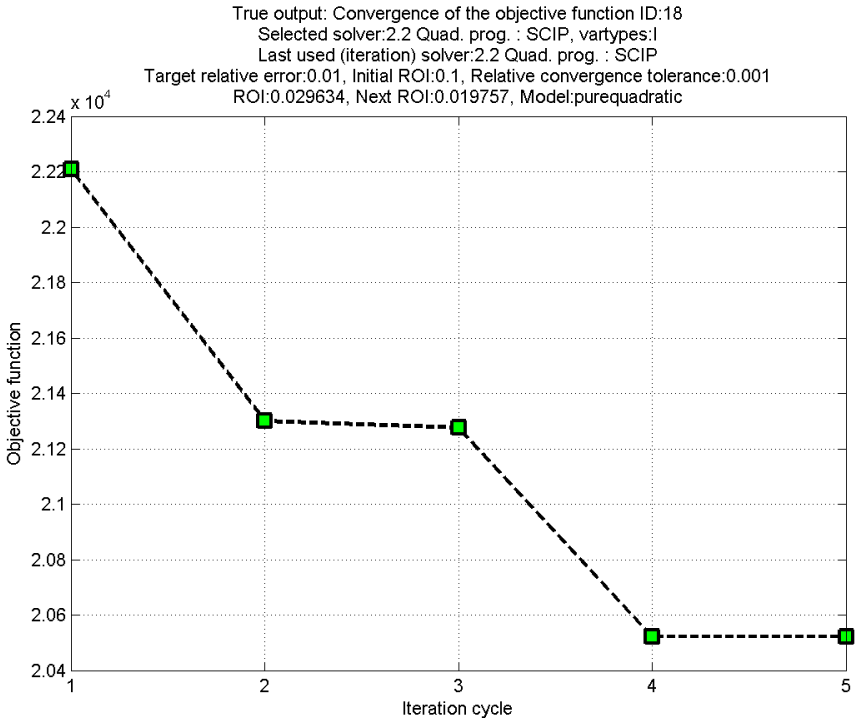
Taulukosta 51 nähdään ennakko-odotuksesta poiketen, että kokonaislukuoptimin hakeminen binäärioptimoinnilla on toiminut tällä kertaa erinomaisesti. Sillä on saatu yhtä hyvä tulos kuin kokonaan kokonaislukutehtävänä ajetulle lineaariselle ja puhtaasti kvadraattiselle tehtävälle. Yllättävää on se, että kvadraattisella mallilla on saatu tässä huonoin tulos. Suunnittelumuuttujia tarkastellessa huomataan vain yhden muuttujan ero siinä verrattuna muihin ajoihin.

Taulukko 51 *Lastiluukun optimoinnin tulokset kokonaislukutehtävänä (algoritmien vertailu).*

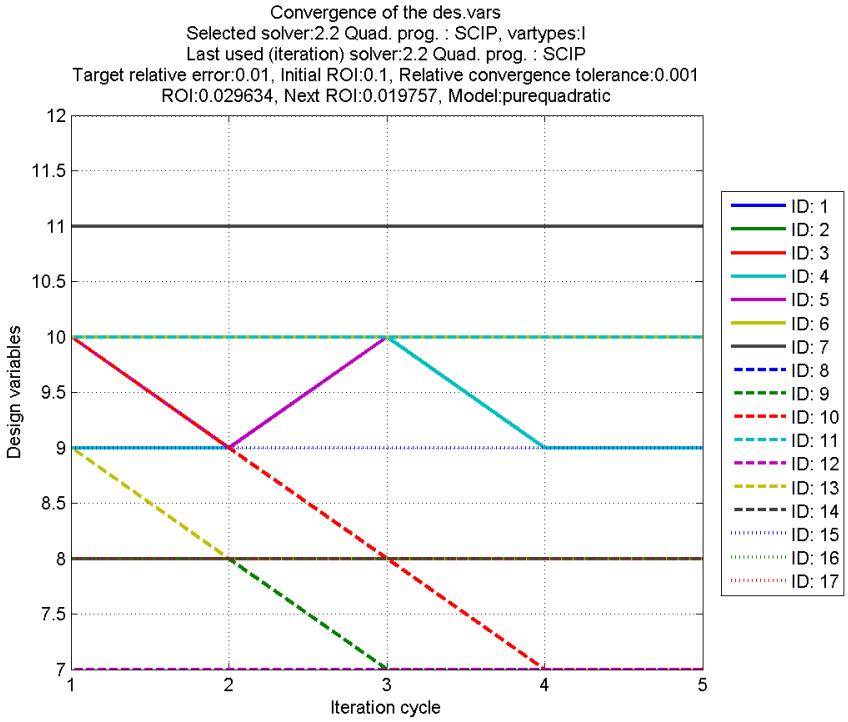
Tunniste	Bufo 28	Bufo 29	Bufo 30	Bufo 31
Metamalli	Lin.	Lin.	Kvadr.	Kvadr.
Algoritmi	ML IP + DH	Gurobi	Scip	Scip
Malli	L	L	Q	PQ
Suunnittelumuuttujat	9 8 9 9 10 10 11 8 7 7 10 7 8 8 9 12 8	9 8 9 9 10 10 11 8 7 7 10 7 8 8 9 12 8	9 8 9 9 10 **11** 11 8 7 7 10 7 8 8 9 12 8	9 8 9 9 10 10 11 8 7 7 10 7 8 8 9 12 8
Tarkka arvo	20521.46	20521.46	20602.99	20521.46
Mallin antama arvo	20521.46	20521.46	20602.99	20521.46
ROI:n säde %	-	0.75	10.27	4.44
Residuaali	0	-0.000003	0.000015	0
Virhe %	0	0 (<1e-6)	0 (<1e-6)	0
Rajoiteyht. keskim. virhe	5e-5	5e-5	0	5e-5
Rajoiteyht. max virhe	0.001799	0.001799	0	0.001799
Käypä	kyllä	kyllä	kyllä	kyllä
Käypien pisteiden lkm	9	13	5	5
Iteraatio	9/9	10/13	4/5	4/5
Aika	44 min	46 min	61 min	27 min

Kvadraattisella optimoinnilla löytyi täysin käypä ratkaisu, mutta muiden ratkaisut olivat myös reilusti annetun virhetoleranssin sisällä ja siten hyväksyttäviä ratkaisuja. Parhaana näistä voidaan pitää puhtaasti kvadraattisella mallilla saatua ratkaisua, koska sillä päästiin hyvään tulokseen vain viidellä iteraatiolla, joihin meni aikaa vain 27 minuuttia. Kuvissa 32-33 esitetään parhaimman tuloksen kuvaajat. Huomataan, että kohde-

funktio suppenee tehokkaasti myös kokonaislukutehtävänä. Lisäksi havaitaan suunnittelumuuttujien kuvaajista, että muuttujat ovat muuttuneet hyvin tasaisesti.



Kuva 32 *Bufo 31, (kokonaislukutehtävä) kohdefunktion suppeneminen (todellinen vaste).*



Kuva 33 *Bufo 31, (kokonaislukutehtävä) muuttujien suppeneminen.*

Muiden kokonaislukutehtävien kuvaajat nähdään liitteistä 16-17. Jatkuvana ensin optimoidun tehtävän (Bufo 28) kohdefunktion kuvaajasta nähdään selvästi kuinka jatkuvasta optimista hypätään kokonaislukuoptimiin.

Lastiluukkutehtävän tulosten perusteella, jos tehtävä ei ole kovin epälineaarinen, niin voidaan käyttää suurempaa tavoitevirhettä kaikilla malleilla tai kiinteää ROI:n sädettä. Puhtaasti kvadraattisen mallin todettiin olevan tässä tehtävätyypissä tehokas ja myös tarkin, kun siinä käytettiin kiinteää 10 % ROI:n sädettä. Tehtävissä saatiin useilla asetuksilla hyviä tuloksia aikaan. Lastiluukkutehtävän ajaminen kokonaislukutehtävänä onnistui myös hyvin.

5.3.2 Dynaaminen palkki

Lopuksi esitellään vielä dynaamisen palkin optimoinnin tulokset. Tämä FEM-malli on tehty työn ohjaajan toimesta testaamista varten. Vasteina tehtävässä ovat massa, jännitys (equilevant stress) ja deformaatio (directional deformation), joista massaa minimoidaan ja muut toimivat rajoitteina.

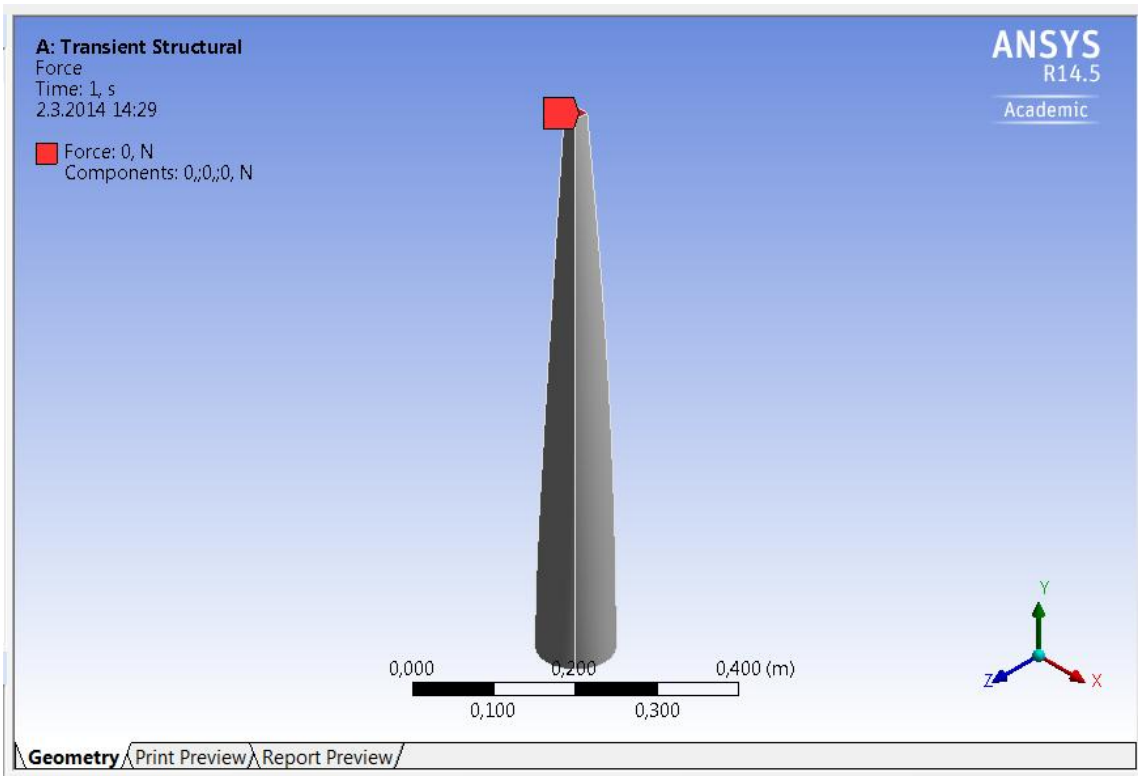
Kuvasta 34 nähdään tehtävän parametrit ANSYS-ohjelmassa. Ohjelmaan ei pidä optimointia varten antaa mitään, koska pisteet lähetetään aina ohjelmaan analysoitavaksi. Tähän on syötetty parametreihin tarkoituksella optimoinnissa käytetyt alkuarvot. Tehtävässä on kahdeksan suunnittelumuuttujaa, joiden ID:t syötetään käyttöliittymän ”Design variables”-kenttään. Vastaavasti annetaan vasteiden ID:t niihin tarkoitettuihin syöttökenttiin.

Outline: No data

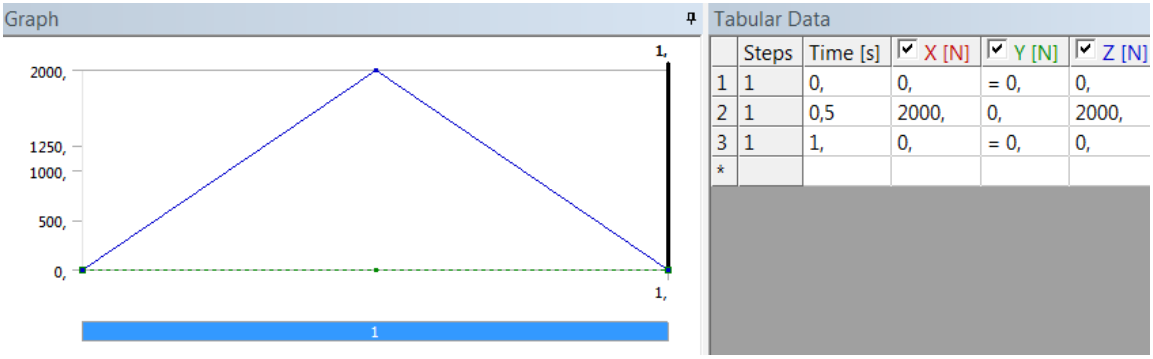
	A	B	C	D
1	ID	Parameter Name	Value	Unit
2	Input Parameters			
3	Transient Structural (A1)			
4	P1	ds_x1@Sketch5@top_opt_140813.Part	40	
5	P2	ds_x2@Sketch8@top_opt_140813.Part	40	
6	P3	ds_x3@Sketch9@top_opt_140813.Part	40	
7	P4	ds_x4@Sketch10@top_opt_140813.Part	40	
8	P5	ds_a1@Sketch5@top_opt_140813.Part	90	
9	P6	ds_a2@Sketch8@top_opt_140813.Part	90	
10	P7	ds_a3@Sketch9@top_opt_140813.Part	90	
11	P8	ds_a4@Sketch10@top_opt_140813.Part	90	
*	New input parameter	New name	New expression	
13	Output Parameters			
14	Transient Structural (A1)			
15	P10	Directional Deformation Maximum	0,84056	mm
16	P9	Equivalent Stress Maximum	62,788	MPa
17	P11	Geometry Mass	25,493	kg
*	New output parameter		New expression	
19	Charts			

Kuva 34 *ANSYS-mallin parametrit.*

Kuvassa 35 nähdään optimoitavan mallin muoto alkuparametreilla. Kuvaan on merkitty myös voima, joka palkkiin kohdistetaan. Kuvasta 36 nähdään, miten tämä kuormitus muuttuu ajan funktiona.



Kuva 35 *Optimoitava malli aloitusparametreilla.*



Kuva 36 *Kuormitus ajan funktiona.*

Tehtävää optimoitiin jatkuvana ja kokonaislukutehtävänä. Taulukosta 52 nähdään käytetyt yleiset asetukset ja lähtötiedot.

Taulukko 52 *Palkin optimoinnissa käytetyt yleiset asetukset ja lähtötiedot.*

Faktoreiden lukumäärä ja tyypit	8 jatkuvaa tai 8 kokonaislukumuuttujaa
Rajoitusyhtälöiden lukumäärä	2
Minimoitava vaste $f(\mathbf{x})$	Massa
Aloituspiste $\mathbf{x}_0$	(40, 40, 40, 40, 90, 90, 90, 90)
Aloituspiste käypä	kyllä
Vasteet aloituspisteessä P11 = f, P9 = g_1, P10 = g_2	$f(\mathbf{x}_0)$ = 25.492965 $g_1(\mathbf{x}_0)$ = 62.787660 $g_2(\mathbf{x}_0)$ = 0.840562
Rajoitusyhtälöiden oikea puoli	100 1.5
Globaalit rajat	$x_1 \ldots x_4 \in [20, 60]$ $x_5 \ldots x_8 \in [45, 135]$
Lopetusehto (konvergenssi)	0.1 %
Maksimi iteraatiot	Lineaariselle 20 Kvadraattisille 10 Palkki 5 asetettu 20.
Aloitus ROI	10 %
Adaptiivinen ROI	kyllä
Suunnittelumuuttujien tarkkuus	1e-4
Tavoitevirhe (TolRel)	1 %
TolCon	1e-2
Techila käytössä	ei

Palkin massaa minimoitiin kolmella eri algoritmilla. Seuraavaan taulukkoon 53 on listattu ajetut optimointitehtävät. Taulukosta nähdään myös kaikkien ajojen tulokset. Tuloksia on analysoitu lyhyesti liitteessä 18. Liitteistä 19-21 nähdään vasteiden ja suunnittelumuuttujien kuvaajat iteraatioiden aikana. Tästä tehtävästä esitetään myös rajoitteina toimivien vasteiden kuvaajat, koska niitä on vain kaksi.

Taulukko 53 *Palkin optimointiajot.*

Tunniste	M	Algoritmi	Kuvaus	Massa	Käypä k/e (virhe)	Aika / h, min	Iter
Palkki 1	L	ML IP	Katso taulukko 52.	14.141742	k (8e-3)	8 h, 21 min	11/20
Palkki 2	L	ML IP	Palkki 1, kiinteä ROI	14.091483	k (9e-3)	4 h, 58 min	15/16
Palkki 3	Q	Scip	Katso taulukko 52.	13.980928	k (8e-5)	11 h, 16 min	9/10
Palkki 4	Q	Scip	Palkki 3, kiinteä ROI	13.939998	k (8e-3)	11 h, 30 min	10/10
Palkki 5	PQ	Scip	Katso taulukko 52.	14.077802	k (3e-3)	5 h, 57 min	12/13
Palkki 6	PQ	Scip	Palkki 5, kiinteä ROI	14.599067	k (6e-3)	3 h, 57 min	7/9
Palkki 7	L	ML IP + DH	Palkki 1, $\mathbf{x} \in \mathbb{Z}$	14.137428	k (0)	3 h, 3 min	12/12
Palkki 8	L	Gurobi	Katso taulukko 52. Kiinteä ROI. $\mathbf{x} \in \mathbb{Z}$	14.108922	k (0)	3 h, 54 min	15/15
Palkki 9	Q	Scip	Palkki 4, $\mathbf{x} \in \mathbb{Z}$	14.0206488	k (0)	9 h, 51 min	8/8
Palkki 10	PQ	Scip	Palkki 6, $\mathbf{x} \in \mathbb{Z}$	14.075106	k (0)	4 h, 31 min	10/10
Palkki 11	L	ML IP + DH	Palkki 2, $\mathbf{x} \in \mathbb{Z}$	14.096157	k (0)	3 h, 6 min	12/12
Palkki 12	L	Gurobi	Katso taulukko 52. $\mathbf{x} \in \mathbb{Z}$	14.029738	k (0)	3 h, 56 min	15/15
Palkki 13	Q	Scip	Katso taulukko 52. $\mathbf{x} \in \mathbb{Z}$	13.904251	k (9.8e-3)	11 h, 37 min	10/10
Palkki 14	PQ	Scip	Katso taulukko 52. $\mathbf{x} \in \mathbb{Z}$	14.143453	k (7e-3)	4 h, 31 min	10/10

Tuloksista huomataan heti, että kaikissa tehtävissä on löydetty virhetoleranssin sisällä oleva ratkaisu. Tehtäviin kulunut aika on myös hyvin suuri, johtuen siitä, että tehtävät jouduttiin ajamaan paikallisesti.

Kuusi ensimmäistä tehtävää on ajettu jatkuvilla muuttujilla ja loput kokonaislukutehtävänä. Taulukossa 54 on tarkemmat tulokset jatkuvista tehtävistä. Taulukkoon on merkitty myös täysin käypä ratkaisu vertailua varten. Laskennan tarkkuudesta ja ratkaisun käypyydestä kerrotaan luvussa 4.1.3.

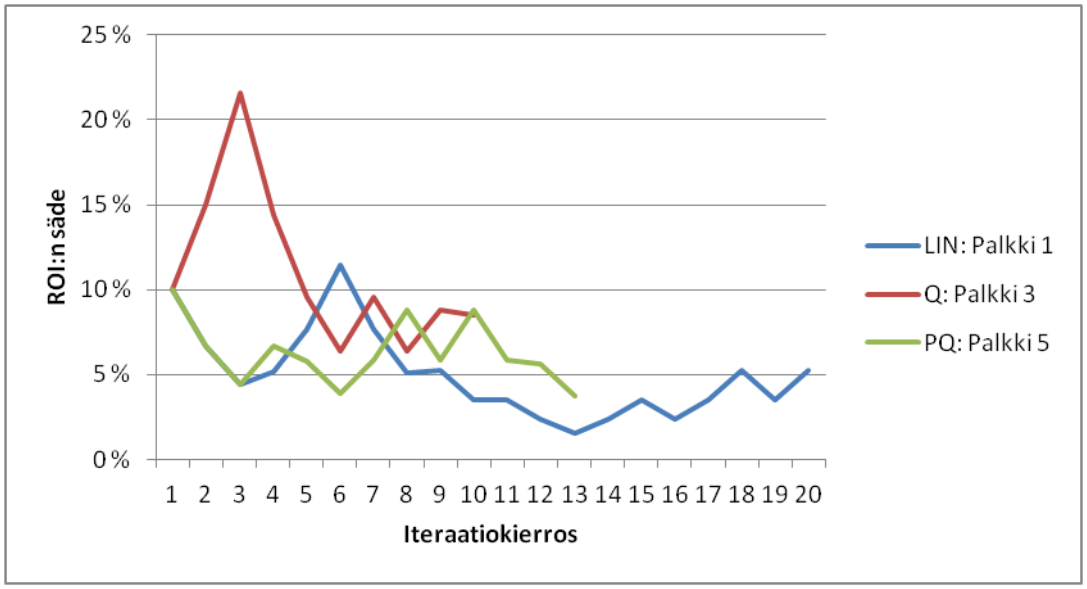
Taulukko 54 *Palkin optimoinnin tulokset (algoritmien, mallien ja ROI:n vertailu).*

Tunniste	Palkki 1	Palkki 2	Palkki 3	Palkki 4	Palkki 5	Palkki 6
Algoritmi	ML IP	ML IP	Scip	Scip	Scip	Scip
Malli	L	L	Q	Q	PQ	PQ
Adaptiivinen ROI	kyllä	ei	kyllä	ei	kyllä	ei
Suunnittelu-muuttujat	38.4934 43.0950 20.0000 20.0000 74.4497 78.2495 45.0000 45.0000	29.4693 46.8221 20.0000 20.0000 102.5272 68.6341 45.0000 45.0000	22.1775 55.6188 20.0000 20.0000 98.6790 62.2401 45.0000 45.0000	23.4864 54.6328 20.0000 20.0000 110.8002 45.0000 45.0000 45.0000	38.8840 40.3496 20.0000 20.0000 85.9189 72.9819 45.0000 45.0000	41.2373 37.9276 21.2576 21.2576 75.7495 79.3098 47.8297 48.4478
Tarkka arvo	14.141742	14.091483	13.980928	**13.939998**	14.077802	14.599067
Mallin antama arvo	14.156267	14.054661	13.979487	13.953468	14.061439	14.602693
ROI:n säde %	3.48	10	6.40	10	5.8641	10
Residuaali	0.014525	-0.036822	-0.001440	0.013470	-0.016363	0.003626
Virhe %	0.102710	0.261308	0.010301	0.096625	0.116236	0.024836
Rajoiteyht. keskim. virhe	0.007508	0.008546	0.000040	0.008203	0.002869	0.006017
Rajoiteyht. max virhe	0.015015	0.017093	0.000080	0.016405	0.005739	0.012034
Iteraatio	11/20	15/16	9/10	10/10	12/13	7/9
Käypä	kyllä	kyllä	kyllä	kyllä	kyllä	kyllä
Käypien pisteiden lkm	20	12	8	10	11	4
Täysin käypien pisteiden lkm (virhe <1e-12)	12	10	5	4	7	2
Tarkka arvo (paras täysin käypä)	14.186973 (iter 18)	14.102350 (iter 11)	13.994885 (iter 10)	13.987114 (iter 9)	14.142621 (iter 11)	21.227439 (iter 2)
Aika	8 h, 21 min	4 h, 58 min	11 h, 16 min	11 h, 30 min	5 h, 57 min	3 h, 57 min

Tuloksista huomataan yhteistä siinä, että usean suunnittelumuuttujan arvoksi on tullut kokonaisluku, vaikka tehtävää optimoitiin jatkuvana. Optimipisteessä kaikki mallit ovat approksimoineet hyvin vastetta. Mielenkiintoista on myös se, että hyvin erilaisilla muuttujien kombinaatioilla on päästy lähes samaan kohdefunktion minimiarvoon.

Tuloksista havaitaan, että lineaarisella mallilla ja adaptiivisella ROI:n säteellä on iterointi jatkunut maksimimäärään asti. Tarkastelemalla liitteiden kuvaajia huomataan, että lineaarisella mallilla tehtävä suppenee aluksi tasaisesti, kunnes ilmenee värähtelyä. Rajoitusyhtälöinä toimivien vasteiden värähtely on vielä voimakkaampaa. Kuvaajista huomataan myös, että mallin antama vaste poikkeaa todellisista vasteista, vaikka vasteet ovat virhetoleranssin sisällä. Selitys värähtelyn alkamiselle, selviää tarkastelemalla

ROI:n säteen käyttäytymistä iteraatiokierroksilla. Tämä nähdään kuvasta 37. ROI:n säde kasvaa lineaarisella mallilla kierroksella kuusi liian isoksi, jolloin tämä aiheuttaa vasteeseen värähtelyn. Värähtelystä huolimatta päästää hyvään tulokseen. Paras ratkaisu löytyi jo kierroksella 11.



Kuva 37 *Palkkitehtävän ROI:n säteen muuttuminen iteraatiokierrosten aikana kun tavoitevirhe TolRel=1 %.*

Lineaarisella mallilla ja kiinteällä ROI:n säteellä suppenee puolestaan ilman värähtelyä ja lopettaa iteroinnit aikaisemmin. Mallin antamat kohdefunktion vasteet vastaavat melko hyvin todellisia vasteita, silti virhetoleranssin alle jääneitä pisteitä on selvästi vähemmän. Tulos on kuitenkin hieman parempi kuin adaptiivisen ROI:n ajossa.

Kuvasta 37 nähdään hyvin kuinka tarkka kvadraattinen malli on tässä tehtävässä adaptiivisella ROI:n säteellä, sillä sädettä kasvatetaan alussa yli 20 %, jonka jälkeen se laskee lähtötasolle. Tuloksista huomataankin, että kvadraattisella mallilla tehtävä suppenee todella tehokkaasti, sillä kuten liitteen kuvaajista nähdään, ollaan lähellä optimia jo kuuden kierroksen jälkeen, mutta iterointi pysähtyy vasta kierroksella 10. Kvadraattinen malli näyttäisi approksimoivan hyvin kohdefunktion lisäksi rajoitefunktioita, koska niissäkään ei näy vastaavaa värähtelyä kuin oli lineaarisen mallin kanssa. Suunnittelumuuttujien kuvaajistakin nähdään tasainen suppeneminen.

Kvadraattisen mallin voidaan olettaa toimivan edellisen perusteellakin hyvin myös kiinteällä ROI:n säteellä. Kiinteällä ROI:n säteellä päästään myös yhtä hyvään tulokseen, mutta suppenee aluksi hieman hitaammin, koska sädettä ei kasvateta. Kiinteällä ROI:n säteellä pysytään asetetun virhetoleranssin sisällä, kun taas adaptiivisella säteellä tuli kaksi ylitystä.

Puhtaasti kvadraattisella mallilla ja adaptiivisella ROI:n säteellä approksimaatiovirhettä tulee enemmän kuin kvadraattisella. Kuvasta 37 nähdään tämä hyvin. ROI:n säde ei kasva yli lähtötason. Puhtaasti kvadraattisella mallilla kohdefunktio suppenee tasai-

sesti ilman värähtelyä. Myöskään rajoitefunktioiden vasteissa ei nähdä yhtä voimasta värähtelyä kuin lineaarisella mallilla. Tällä päästään hyvään tulokseen.

Kiinteällä ROI:n säteellä puhtaasti kvadraattisella mallilla kohdefunktio suppenee vielä tehokkaammin, mutta virhetoleranssia rikotaan useammalla kierroksella, myös kahden viimeisen kierroksen kohdalla. Tästä syystä parhaaksi ratkaisuksi valitaan kolmanneksi viimeisen kierroksen tulos, joka on tässä vertailussa hieman muita huonompi.

Tuloksista huomataan myös, että näissä tehtävissä löydetään iteraatioiden aikana täysin käypiä ratkaisuja varsin paljon, ottaen huomioon kuinka tarkkoja ratkaisujen täytyy silloin olla. Olennainen huomio on myös se, että esimerkiksi Palkki 6 -ajon kanssa, täysin käypä ratkaisu on siinä huomattavasti huonompi kuin sallitun virhetoleranssin sisällä oleva tulos. Tästä syystä ei voida käytännössä vaatia näin tarkkoja tuloksia, ellei sitten oikeasti ole tarvetta sille. Asetetun virhetoleranssin idea on, että käyttäjä voi asettaa omaan tehtävään hyväksyttävän virhetason, joka riippuu paljon tehtävässä. Tähän tehtävään asetettiin virhetoleranssiksi 1e-2, jolla saatiin myös hyviä tuloksia.

Lopuksi tehtävät ajettiin vielä kokonaislukutehtävinä. Taulukossa 55 esitetään tarkemmin tulokset vain adaptiivisella ROI:n säteellä saaduista. Kaikkien ajojen tulokset nähdään taulukosta 53 ja lisäksi liitteistä.

Taulukko 55 *Palkin optimoinnin tulokset kokonaislukutehtävänä (algoritmien vertailu).*

Tunniste	TPalkki 7	TPalkki 12	TPalkki 13	TPalkki 14
Metamalli	Lin.	Lin.	Kvadr.	Kvadr.
Algoritmi	ML IP + DH	Gurobi	Scip	Scip
Malli	L	L	Q	PQ
Suunnittelumuuttujat	40 42 20 20 80 71 45 45	24 54 20 20 95 67 45 45	20 59 20 20 60 86 50 45	41 37 20 20 79 87 45 45
Tarkka arvo	14.137428	14.029738	13.904251	14.143453
Mallin antama arvo	-	14.013163	13.906438	14.130190
ROI:n säde %	-	5.02	1.2368	6.4357
Residuaali	-	-0.016575	0.002187	-0.013262
Virhe %	-	0.118142	0.015726	0.093771
Rajoiteyht. keskim. virhe	0	0	0.009807	0.006581
Rajoiteyht. max virhe	0	0	0.019614	0.013163
Iteraatio	12/12	15/15	10/10	10/10
Käypä	kyllä	kyllä	kyllä	kyllä
Käypien pisteiden lkm	11	13	8	9
Täysin käypien pisteiden lkm (virhe <1e-12)	9	12	3	5
Tarkka arvo (paras täysin käypä)	14.137428 (iter 12)	14.029738 (iter 15)	14.159593 (iter 6)	15.014390 (iter 8)
Aika	3 h, 3 min	3 h, 56 min	11 h, 37 min	4 h, 31 min

Kokonaislukutehtävissä mallit ovat myös approksimoineet hyvin vastetta optimipisteessä. Vastaavasti kuin jatkuvissa tehtävissä, niin on tässäkin päästy hyvin erilaisilla muuttujien kombinaatioilla lähes samaan kohdefunktion minimiarvoon.

Lineaarisella mallilla ja adaptiivisella ROI:n säteellä löydetään jatkuvan optimin jälkeen hyvä tulos binäärioptimoinnilla, vastaava tulos kuin muissa kokonaislukutehtävissä. Kaikissa kokonaislukutehtävissä päästään hyviin tuloksiin, vastaavasti kuin jatkuvissa tehtävissä. Lineaarisia kokonaislukutehtäviä laskettiin myös Gurobilla. Tuloksista nähdään, että silläkin päästiin hyvää tulokseen. Gurobin kanssa tulee vain samanlaista värähtelyä kuin tuli lineaarisella mallilla jatkuvassa tehtävässä.

Kvadraattiset mallit toimivat kokonaislukutehtävissäkin tarkemmin, jos verrataan mallin antamia ja todellisia vasteita, mutta kuten sanottua kaikilla malleilla päästiin loppujen lopuksi hyvään tulokseen.

Tehokkuusmielessä kvadraattinen malli adaptiivisella ROI:n säteellä oli todella huono, jos katsotaan tehtävään kulunutta aikaa. Iteraatioiden määrässä se oli yhtä hyvä kuin puhtaasti kvadraattinen. Tarkimpaan tulokseen kokonaislukutehtävissä päästiin kuitenkin kvadraattisella mallilla ajossa TPalkki 13. Tämän tehtävän olisi saanut varsin nopeasti ajettua Techila-laskentaympäristöllä, jos sen olisi saanut toimimaan tällä FEM-mallilla.

Kaikissa dynaamisen palkin ajoissa rikottiin korkeintaan deformaatiolle asetettua ylärajaa. Jännitys pysyi kaikissa tehtävissä reilusti alle ylärajan.

Diskreetin pisteen haussa ajoissa Palkki 7 ja Palkki 11 tehtiin mielenkiintoinen havainto, että vasteet eivät täysin vastanneet Palkki 1 ja Palkki 2 ajossa saatuja tuloksia, vaikka ensin mainituissa haetaan lopuksi vain kokonaislukuoptimi binäärioptimoinnilla. Ilmeisesti dynaamisen mallin kanssa voi tulla pientä eroa analyysien kanssa. Lastiluukkumallin kanssa vastaavaa ei huomattu.

Päätettiin tutkita jännityksen käyttäytymistä tarkemmin. Tutkittiin asiaa koepisteellä (36, 36, 36, 36, 81, 81, 81, 81), joka oli toisella kierroksella saatu optimipiste ajoissa Palkki 1 ja 7. Tehtiin lisäksi kolme testiä suoraan ANSYS-ohjelmalla, ajamalla analyysi uudestaan samalla koepisteellä. Taulukosta 56 nähdään saadut tulokset.

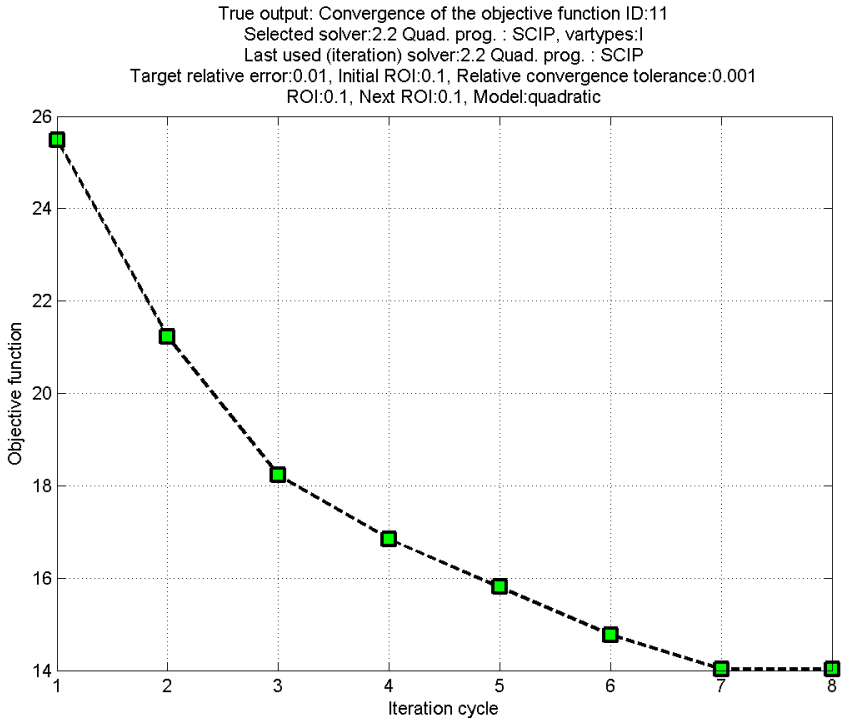
Taulukko 56 *Dynaamisen mallin tarkkuuden tarkastelu.*

Lähde	Jännitys koepisteessä (36, 36, 36, 36, 81, 81, 81, 81)
Palkki 1 -ajo	68.5633**7**0163439500
Palkki 7 -ajo	68.5633**7**0630232910
ANSYS testi 1	68.5633**6**7371474740
ANSYS testi 2	68.5633**6**8518466959
ANSYS testi 3	68.5633**6**7728147227

Huomataan, että samoilla muuttujien arvoilla ANSYS palauttaa hieman eri tuloksia. Ero on hyvin pieni. Palkki-tehtävien tuloksia tarkemmin tarkastelemalla huomataan, että myös mallin ja tarkan arvon välillä on eroa näiden ajojen kesken. Tällöin myös adaptiivisella ROI:n säteellä kohdefunktio suppenee hieman eri lailla. Erot kertautuvat iteraatiokierroksilla, jolloin vasteiden suppeneminen myös poikkeaa eri ajoissa. Suhteelliset virheet lasketaan MATLABin tarkoilla arvoilla, joten eroa tulee helposti jos todelliset vasteet poikkeavat eri ajoilla. Lopputulos on kaikilla ajoilla kuitenkin hyvä, mutta on hyvä tietää, että vastaavilla ajoilla voi tulla eroa tästäkin syystä.

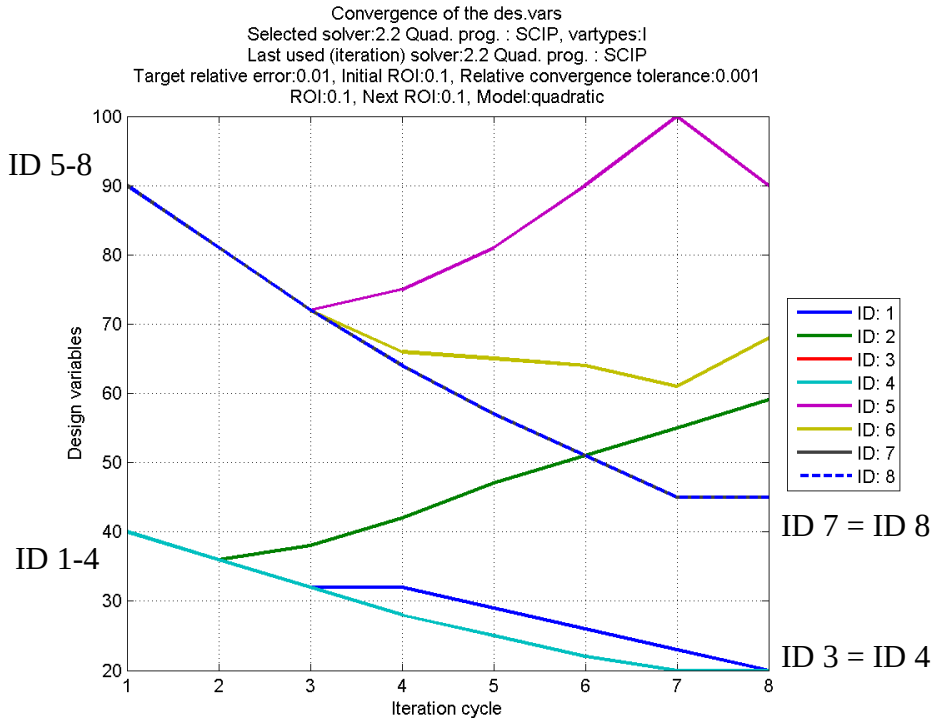
Toinen syy eri tuloksiin samoilla laskentatyökalun parametreilla voi johtua koepisteiden laskennasta. MATLABin D-optimaaliseen suunnitteluun tarkoitetut funktiot eivät aina tarjoa optimaalista suunnittelua (Matlab 2013). Edellä kuvattu ero ei kuitenkaan johtunut koesuunnittelusta vaan ANSYS-ohjelman palauttamista poikkeavista vasteiden arvoista.

Parhaaksi tulokseksi kokonaislukutehtävistä valittiin Palkki 9 -ajo, koska sillä päästiin hyvään kohdefunktion arvoon rikkomatta yhtään rajoitusyhtälöitä. Kuvasta 38 nähdään tämän ratkaisun kohdefunktion eli massan suppeneminen. Huomataan, että puhtaasti kvadraattisella mallilla ja kiinteällä 10 % ROI:n säteellä tehtävä suppenee tehokkaasti ja tarkasti.



Kuva 38 *Palkki 9 (kokonaislukutehtävä), kohdefunktion suppeneminen (todellinen vaste).*

Kuvasta 39 nähdään vielä miten suunnittelumuuttujat ovat supenneet. Kuvaajasta huomataan, että useat muuttujat saavat samoja arvoja kierrosten aikana.



Kuva 39 *Palkki 9 (kokonaislukutehtävä), muuttujien suppeneminen.*

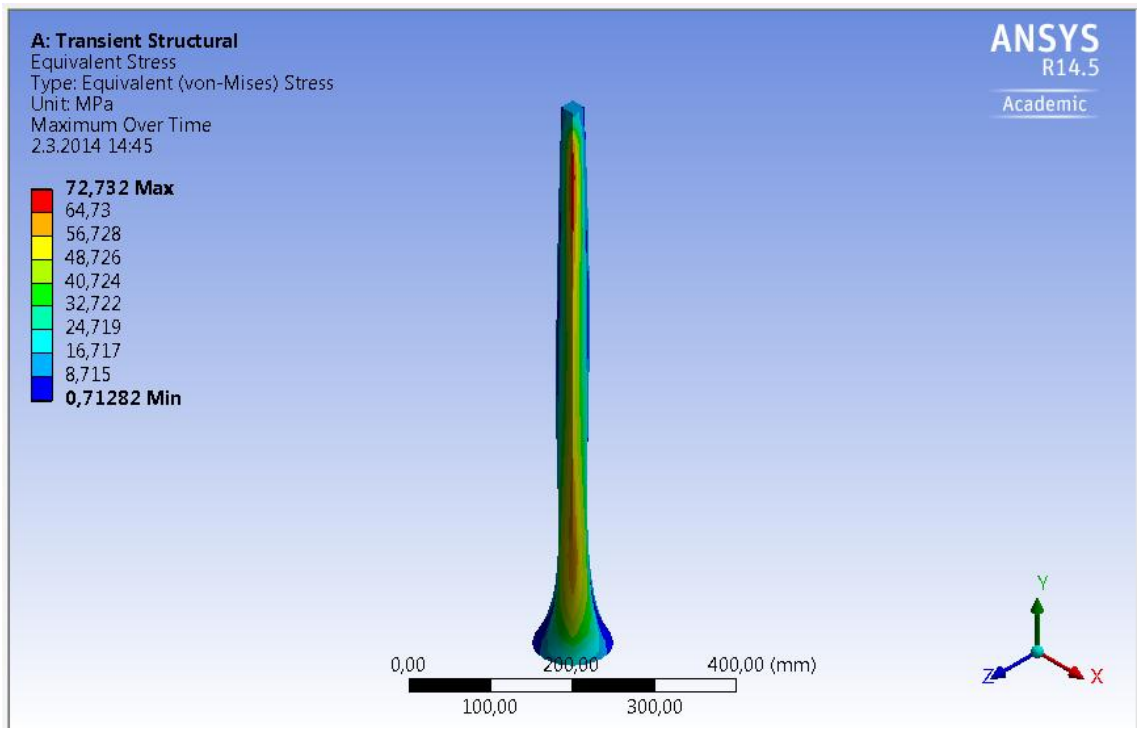
Kuvasta 40 nähdään valitun kokonaislukuratkaisun parametrit ANSYS-ohjelmassa analysoituna. Huomataan, että löydettiin täysin käypä rakenne, jonka massa pieneni 25.5 kg:sta 14 kg:aan.

Outline: No data

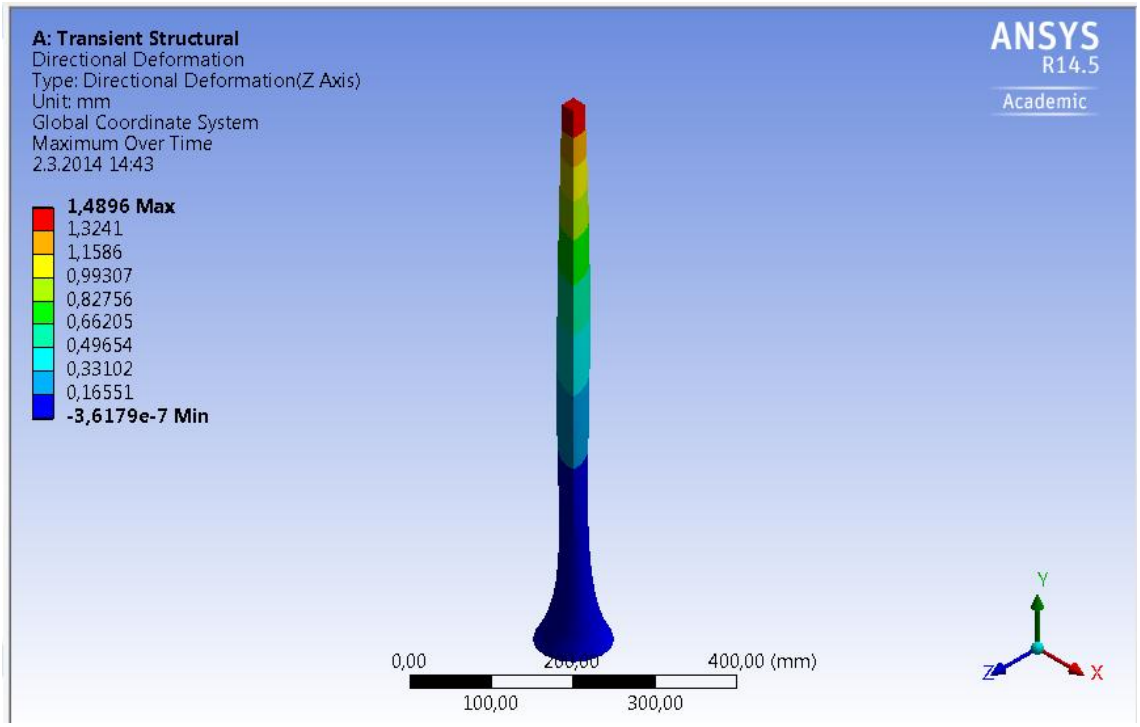
	A	B	C	D
1	ID	Parameter Name	Value	Unit
2	Input Parameters			
3	Transient Structural (A1)			
4	P1	ds_x1@Sketch5@top_opt_140813.Part	20	
5	P2	ds_x2@Sketch8@top_opt_140813.Part	59	
6	P3	ds_x3@Sketch9@top_opt_140813.Part	20	
7	P4	ds_x4@Sketch10@top_opt_140813.Part	20	
8	P5	ds_a1@Sketch5@top_opt_140813.Part	90	
9	P6	ds_a2@Sketch8@top_opt_140813.Part	68	
10	P7	ds_a3@Sketch9@top_opt_140813.Part	45	
11	P8	ds_a4@Sketch10@top_opt_140813.Part	45	
*	New input parameter	New name	New expression	
13	Output Parameters			
14	Transient Structural (A1)			
15	P10	Directional Deformation Maximum	1,4896	mm
16	P9	Equivalent Stress Maximum	72,732	MPa
17	P11	Geometry Mass	14,021	kg
*	New output parameter		New expression	
19	Charts			

Kuva 40 *Optimoidun mallin parametrit ANSYS-ohjelmassa.*

Kuvista 41 ja 42 nähdään ANSYS-ohjelman esittämänä saatu optimaalinen muoto tarkastelemalla erikseen jännitystä ja deformaatiota.



Kuva 41 *Optimoidun mallin muoto ja jännitys (1. rajoitusyhtälö).*



Kuva 42 *Optimoidun mallin muoto ja deformaatio (2. rajoitusyhtälö).*

Dynaamisen palkin optimointi onnistui yli odotusten. Kohteena olevaa massaa saatiin reilusti minimoitua. Seuraavaksi esitetään yhteenveto saaduista tuloksista.

5.4 Yhteenveto

Tulosten perusteella voidaan todeta kuinka tärkeää yleisesti on kehitetyn laskentamenetelmän tarkkaan mietitty testaaminen erityyppisillä käytännön tarpeita vastaavilla tehtävillä. Suppealla testijoukolla voidaan päätyä täysin eri johtopäätökseen kuin huolella mietityllä kokoelmalla erityyppisiä tehtäviä, joita kokeillaan menetelmään liittyvillä keskeisillä asetuksilla. Tässä työssä voidaan tulosten perusteella testiasettelun olleen tarpeeksi kattava, sillä erilaisia asetuksia kokeilemalla erityyppisten testiprobleemien ja FEM-rakennemallien optimointiin, saatiin tehtyä useita mielenkiintoisia havaintoja.

Kaikkien edellä esitettyjen tehtävien optimointi meni erittäin hyvin, kun oli valittu sopivat asetukset laskentatyökaluun. Keskeisiä laskentatyökalun asetuksia ovat mallin ja algoritmin valinnan lisäksi, tavoitevirhe eli sallittava approksimaatiovirhe ja ROI:n alkusäde sekä se, että käytetäänkö adaptiivista ROI:n sädettä.

Optimoitujen tehtävien tärkein havainto oli se, että adaptiivisella ROI:n säteellä ROI:n säde voi mennä hyvin pieneksi, jolloin suppeneminen radikaalisti hidastuu. Tämä huomattiin esimerkiksi tehtävissä hitsattu palkki (HP), nopeudenalennin (SR) ja TP113. Lisäksi huomattiin, että erityisesti kierrejousen ja nopeudenalentimen optimoinnissa kiinteällä ROI:n säteellä rikottiin asetettua virhetoleranssia selvästi useammin.

Asetettavalla tavoitevirheellä voidaan siis vaikuttaa siihen kuinka helposti kasvatetaan ROI:n sädettä, jonka avulla lasketaan uuden aliavaruuden rajat. Käytännössä sopivaa tavoitevirheen arvoa voi olla kuitenkin vaikeaa löytää. Lisäksi, erittäin epälineaarisissa tehtävissä se pitäisi asettaa hyvin suureksi, josta voi taas aiheutua helposti liian suurta approksimaatiovirhettä. Tämä näkyy yleensä vasteiden värähtelynä ja kohdefunktion suppenemisen huonontumisena.

Nykyisessä ratkaisussa voidaan tulosten perusteella suositella epälineaariselle tehtävälle 1 % tavoitevirhettä ja 10 % ROI:n alkusädettä kaikille metamalleille. Kvadraattisilla malleilla voidaan käyttää usein hieman suurempaakin tavoitevirhettä. Suuremmalla tavoitevirheellä saadaan kohdefunktio suppenemaan nopeammin, mutta riskinä on edellä mainittu vasteiden värähtely ja virhetoleranssin rikkominen.

Sopivan metamallin valintaan auttaa jos käyttäjällä on tietoa optimoitavan kohteen vasteiden käyttäytymisestä. Tulosten analysoinnissa havaittiin, että tehtävän ristikkäistermit hankaloittavat vasteiden approksimointia. Tällöin adaptiivisella ROI:n säteellä suppeneminen voi hidastua merkittävästi käytettäessä lineaarista tai puhtaasti kvadraattista mallia. Jos tiedetään tehtävässä olevan paljon muuttujien ristikkäisvaikutuksia, niin parempiin tuloksiin voidaan päästä käyttämällä kvadraattista mallia. Lineaarisella ja puhtaasti kvadraattisella mallilla päästään erittäin epälineaarisissa tehtävissä usein tehokkaaseen suppenemiseen vain käyttämällä kiinteää ROI:n sädettä. Suunnittelussa haetaan siten kompromissia tarkkuuden ja tehokkuuden väliltä.

Tulosten perusteella adaptiivisen ROI:n säteen käyttö on suositeltavaa, mutta sen määrittämistä on syytä kehittää. Tutkimisen arvoisena kehitysideana suositellaan, että approksimaatiovirheen lisäksi voisi käyttää uuden iteraation ROI:n säteen määrittämiseen hyödyksi myös sallittua rajoitusyhtälöiden rikkomista. Jos todellisten vasteiden ratkaisu on sallitun virhetoleranssin sisällä, niin mallin approksimaation voidaan ajatella olevan myös riittävä. Tällöin voitaisiin pitää ROI:n säde vähintään samana. Ratkaisun ollessa virhetoleranssin sisällä, ROI:n sädettä voitaisiin kasvattaa jos myös approksimaatiovirhe on alle tavoitevirheen. Virhetoleranssin ylityksessä käytettäisiin alkuperäistä luvun 4.1.2 kaavaa (38). Tämä ratkaisu voi aiheuttaa suunnittelumuuttujiin värähtelyä, mutta se tarkoittaa myös sitä, että haetaan laajemmin sopivia suunnittelumuuttujia, mikä on hyvä asia. Esitetyssä kehitysideassa asetettu virhetoleranssi siis myös ohjaisi iteratiivista optimointia eikä pelkästään toimisi laskentatyökalun käyttäjän apuna käypien pisteiden valintaan. Tätä esitettyä kehitysideaa ei ehditty työn loppupuolella kuitenkaan kokeilla.

Tulosten analysoinnissa havaittiin, että lineaariset optimointialgoritmit ovat yhtä tarkkoja. Interior point -alogoritmia voidaan niistä suositella, koska se palauttaa vähintään parhaiten käyvän ratkaisun, jolloin laskentatyökalun vaiheittainen optimointi ei keskeydy. Eli sitä suositellaan käytettävän etenkin kun lähdetään epäkäyvästä aloituspisteestä. Lisäksi vaiheittaisessa vastepintamenetelmässä voi tulla iteraatioiden aikana optimointitehtävä, jossa käypä joukko on tyhjä. Tästä syystä esimerkiksi Simplex-algoritmia ei yleensä kannata käyttää laskentatyökalun kanssa, koska se ei selviä siitä.

Kvadraattisista algoritmeista todettiin, että Scip-algoritmilla saadaan hyviä tuloksia aikaan, sillä se osaa ratkaista myös epäkonveksin tehtävän. Tehtävässä, jossa käypä joukko on tyhjä, algoritmi palauttaa virheen. Tällöin laskentatyökalu vaihtaa väliaikaisesti lineaariseen malliin ja optimoi tehtävän uudestaan MATLABin Interior-pointilla. Toisen kvadraattiseen optimointiin tarkoitetun Gurobin algoritmin, huomattiin pystyvän ratkaisemaan vain konvekseja tehtäviä. Tästä syystä Gurobia ei voida suositella kvadraattiseen optimointiin.

Jos tarkoituksena on käyttää kvadraattisia malleja optimointiin eikä tiedetä käypää aloituspistettä, niin käyttäjän on tällöin ensin suositeltavaa hakea sopiva aloituspiste käyttämällä Interior point -algoritmia. Kuten edellä mainittiin, niin laskentatyökalu osaa automaattisesti vaihtaa siihen väliaikaisesti, mutta tällöin on ensin laskettu koepisteet turhaan kvadraattiselle mallille. Lineaaristen algoritmien kanssa ei lasketa koepisteitä uudestaan epäkäyvässä tapauksessa, vaan optimoidaan heti uudestaan Interior point -algoritmilla.

Kokonaislukuoptimoinnilla saatiin myös hyviä tuloksia aikaan. Kvadraattisiin sekalukutehtäviin käytettiin Scip-algoritmia ja lineaarisiin Gurobin algoritmia. Scip-algoritmia olisi työkalussa voitu käyttää myös lineaarisiin tehtäviin, mutta tällä hetkellä se on implementoitu ainoastaan kvadraattisiin tehtäviin. Kokonaislukuoptimoinnissa käytettiin myös adaptiivista ROI:n sädettä. Siinä hieman rikotaan menetelmään liittyvää ideaa, koska adaptiivisella ROI:n säteellä muutettavia rajoja pyöristetään kokonaisluku-

vaatimuksiin. Tästä huolimatta adaptiivinen ROI:n säde toimi kokonaislukutehtävissä yleisesti ottaen hyvin.

Kokonaislukutehtävissä käytettiin myös menetelmää, jossa haetaan ensin jatkuva optimi, josta haetaan sitten binäärioptimoinnilla kokonaislukuoptimi. Tämä toimi pääasiassa hyvin, mutta esimerkeillä TP 12 ja E3 osoitettiin, että sillä ei aina löydetä parasta tavoitettavissa olevaa ratkaisua. Käytännön tehtäviin sen antama tarkkuus, kuitenkin usein riittää.

Työssä laadittuja diskreettejä malleja testattiin muutaman testiprobleeman kanssa. Niiden kanssa saatiin hyviä tuloksia aikaan ja voidaan todeta, että mallit toimivat myös käytännössä. Työkalussa ei kuitenkaan ole otettu huomioon koepisteiden laskentaan liittyviä mahdollisia ongelmia. Käytännön FEM-rakennemallin optimointitehtävissä voi olla kiellettyjä koepisteitä joita ei voida laskea. Laskentatyökalun koesuunnittelussa käytettävällä MATLABin D-optimaalisella suunnittelulla voi tulla koematriisi, jossa on koepisteitä joita ei voida laskea. Laaditut diskreetit mallit eivät tähän vaikuta, sillä niiden avulla vain pakotetaan optimoinnissa valitsemaan parhaiten sopiva kokoonpano. Koematriisia tarvitaan vasteiden laskentaan, joiden avulla saadaan sovitettua metamalli. Täten on tarpeellista selvittää, että miten luodaan koematriisi tehtäviin, joissa on sallittuja kokoonpanoja eli toisistaan riippuvia muuttujia. Tämä voisi onnistua määrittelemällä itse kandidaattimatriisi candexch-funktiolle, jolloin saadaan FEM-rakennemallille varmasti käypä koematriisi (Matlab 2013). Tätä asiaa on suositeltavaa tutkia ja implementoida löydetty ratkaisu työkaluun. Testiprobleemien kanssa ei tullut tämän kanssa ongelmia, koska kaikki koepisteet niissä voitiin laskea.

Tutustumalla simulaatiopohjaiseen optimointiin soveltuviin valmisohjelmiin huomattiin, että niissä on mahdollisuus ketjuttaa algoritmeja (OPTIMUS 2014). Implementoidussa vaiheittaisessa vastepintamenetelmässäkin siitä olisi selvää hyötyä. Esimerkiksi siten, että voitaisiin ensin hakea käypä alue lineaarisella mallilla ja Interior point -algoritmilla, jonka jälkeen vaihdettaisiin kvadraattiseen malliin ja siihen sopivaan algoritmiin. Toinen hyvä tapa ketjuttaa voisi olla käyttää ensin lineaarista mallia sen tehokkuuden takia, kunnes ollaan lähellä optimipistettä, jonka jälkeen vaihdettaisiin kvadraattiseen malliin tarkkuuden saamiseksi. Erilaisia metamallien ja algoritmien ketjuttamisvaihtoehtoja on siten suositeltavaa pohtia ja mahdollisesti implementoida ratkaisut työkaluun. Käyttäjän tulisi pystyä sitten tekemään helposti käyttöliittymän kautta monipuolisia ketjutuksia ja asettaa niille mahdollisesti lisäasetuksia. Yhtenä lisäasetuksena voisi olla se, että millä ROI:n säteellä lähdetään tarkentamaan ratkaisua esimerkiksi kvadraattisella optimoinnilla. Tässä huomataan selvä parannuskohde työkalun toimintaan, jolla saadaan todennäköisesti parempia tuloksia aikaan ja lisäksi ennen kaikkea ideana on se, että käyttäjän ei tarvitsisi ajaa tehtävää uudestaan erilaisilla työkalun parametreilla. Nykyisessä työkalun versiossa on edellä kerrottu vaikeus valita keskeisiä työkalun parametreja. Keskeinen selvä hyöty ketjuttamisella on se, että sen avulla voidaan saada kohdefunktio uudestaan suppenemaan jos se hidastuu tai jää samalle tasolle.

Vastepintamenetelmällä voitaisiin päästä parempiin tuloksiin myös jos haettaisiin parempia suunnittelumuuttujien arvoja jollain menetelmällä myös ROI:n ulkopuolelta, esimerkiksi muutaman kierroksen välein. Jos parempi piste löytyy, niin jatketaan siitä.

Aliavaruuden virittäminen on keskeistä vaiheittaisessa vastepintamenetelmässä. Tästä syystä työkalua varten on hyvä selvittää muita vaihtoehtoja aliavaruuden virittämiseen. Tavat voisivat toimia myös valitun menetelmän rinnalla tai siten, että menetelmää vaihdetaan iterointien aikana.

Laskentatyökalun metamallimenetelmiin kuuluvan vastepintamenetelmän lisäksi työkaluun on mahdollista lisätä myös muita simulaatiopohjaiseen optimointiin tarkoitettuja menetelmiä. Näitä menetelmiä voisi olla hyvä myös tarvittaessa pystyä ketjuttamaan. Esimerkiksi neuroverkon ja sen optimoinnin avulla voitaisiin löytää suuremmasta suunnitteluavaruuden osasta optimi, josta voitaisiin jatkaa esimerkiksi kehitetyllä vastepintamenetelmällä.

Käytännön tehtävissä on tavallisesti useita optimoitavia vasteita, jolloin tarvitaan monitavoiteoptimointiin liittyviä menetelmiä. Laskentatyökalun monitavoiteoptimoinnin kehittämistä ja testaamista on suositeltavaa jatkaa.

Testiprobleemissa päästiin todella lähelle globaalia minimiä. Niiden perusteella, tehtyä työkalua voidaan pitää tarkkana myös erittäin epälineaarisiin tehtäviin. Tehdyistä ANSYS-mallin optimoinneista ei tiedetä kuinka lähelle päästiin globaalia minimiä, mutta kohdefunktion arvo pieneni joka tapauksessa huomattavasti molemmissa tehtävissä. Esimerkiksi tässä työssä tarkasteltavalle dynaamiselle palkille, jota optimoitiin kokonaislukutehtävänä, löydettiin täysin käypä rakenne, jonka massa pieneni 25.5 kg:sta 14 kg:aan. Vaikka laskentatyökalulla saatu optimi ei olisikaan globaali optimi, saadaan sillä silti yleensä merkittävä parannus lähtötilanteeseen.

6 JOHTOPÄÄTÖKSET

Heti alkuun voidaan todeta, että kaikki työlle asetetut tavoitteet täyttyivät. Saatiin kehitettyä laskentatyökalu, jota voidaan käyttää simulaatiopohjaiseen optimointiin lukuisten testien perusteella menestyksekkäästi. Kehitettyyn laskentatyökalun ensimmäiseen versioon on implementoitu Tampereen teknillisen yliopiston Kone- ja tuotantotekniikan laitoksella ideoitu vaiheittaisen vastepintamenetelmän sovellus.

Vaiheittaisen vastepintamenetelmän todettiin olevan metamallina hyvin robusti eli sillä saatiin hyviä tuloksia aikaan vaikka tiedossa ei olisi optimoitavalle FEM-rakennemallille käypiä suunnittelumuuttujia. Ennen kaikkea aina päästiin huomattavasti parempaan lopputulokseen kuin mistä lähdettiin ja ilman keskeytyksiä. Edellytyksenä hyvin onnistuneelle optimoinnille on kuitenkin se, että käyttäjällä on hieman tietoa optimoitavan kohteen vasteiden käyttäytymisestä.

Erittäin epälineaaristen tehtävien kanssa on työssä saatujen tulosten perusteella suositeltavaa käyttää approksimaatiovirheen perusteella muutettavaa tarkasteltavan aliavaruuden kokoa ja kvadraattista mallia. Jos metamallilla saadaan approksimoitua hyvin todellisia vasteita, voidaan ikkunan koko pitää vakiona. Lineaarisella mallilla menetetään usein tarkkuutta, mutta saavutetaan tehokkuutta, koepisteiden laskemisen lukumäärän pysyessä kohtuullisena.

Työkalun vastepintamenetelmän keskeisiä asetettavia parametreja ovat sallittava approksimaatiovirhe ja tarkasteltavan aliavaruuden koko tehtävän alussa. Tässä työssä on esitetty näille kokemukseen perustuvia arvoja, joilla yleensä päästään hyviin tuloksiin. Approksimoitavan mallin tarkkuus vaikuttaa näiden parametrien valintaan. Korkeamman asteen kuten toisen asteen eli kvadraattisella mallilla, voidaan usein käyttää hieman suurempaa ikkunan kokoa ja sallia suurempi approksimaatiovirhe. Sallittava approksimaatiovirhe vaikuttaa siihen kuinka helposti kasvatetaan tarkasteltavaa aliavaruutta seuraavalle iteraatiokierrokselle. Mallin tarkkuuteen vaikuttaa lisäksi erityisesti, miten malli on tuotettu. Eli mikä on käytetty koesuunnittelumenetelmä ja koepisteiden määrä.

Vaiheittaisissa vastepintamenetelmissä tarkkuuteen ja tehokkuuteen vaikuttaa erityisesti se, miten tarkasteltava aliavaruus viritetään. Implementoidussa vastepintamenetelmässä aliavaruus viritetään ottamalla edellinen optimipiste sen keskipisteeksi ja rajat saadaan erikseen laskemalla. Implementoidun vastepintamenetelmän tarkkuutta ja tehokkuutta voidaan merkittävästi parantaa kehittämällä menetelmää, jolla nämä aliavaruuden rajat määritetään. Nykyisellä menetelmällä kohdefunktio suppenee erittäin epälineaarisissa tehtävissä muutaman iteraation jälkeen hyvin hitaasti, koska ikkunan kokoa pienennetään koko ajan, ellei sallittava approksimaatiovirhe ole riittävän suuri. Sopivan tavoitevirheen asettaminen voi olla vaikeaa. Tästä seuraa myös se, että voidaan jäädä

helposti huonompaan arvoon kuin olisi päästy suuremmalla ikkunan koolla. Parannusehdotuksena nykyiseen ratkaisuun on se, että otetaan aliavaruuden rajojen määrittelyssä huomioon myös asetettu virhetoleranssi rajoitusyhtälöiden ylitykselle. Mallin approksimaation voidaan ajatella olevan riittävä myös silloin, kun todellisten vasteiden optimiratkaisu on tämän virhetoleranssin sisällä. Tällöin voitaisiin pitää aliavaruuden koko vähintään samana. Tällä muutoksella kohdefunktio suppenee nopeammin eikä jäisi niin helposti huonompaan arvoon. Lisäksi muitakin ratkaisuja tähän on suositeltavaa selvittää. Kiinteän ikkunan koon käyttäminen ei ole yleensä erittäin epälineaarisissa tehtävissä suositeltavaa, ellei ikkunan koko ole hyvin pieni. Tarkasteltava aliavaruus voidaan virittää myös muulla tavalla kuin implementoidussa ratkaisussa. Näitä muita tapoja on suositeltavaa tutkia lisää ja hyödyntää tutkimustuloksia kehitetyssä laskentatyökalussa.

Työssä huomattiin, että valitulla lineaarisella optimointialgoritmilla ei ollut merkitystä lineaarisen mallin optimoinnin tarkkuuteen ja tehokkuuteen. Tämä johtuu siitä, että käytetyt perinteiset optimointimenetelmät ovat kaikki tehokkaita kun tehtävän koko on pieni. Optimoinnissa tehtävää pidetään yleensä suurena, kun muuttujia on kymmeniä tuhansia tai enemmän. FEM-rakennemallien optimoinnissa suunnittelumuuttujien määrä on yleensä muutamasta muutamaan kymmeneen. Lineaariset jatkuvat optimointitehtävät eivät ole siis tässä käytössä optimointiongelmana algoritmille haastavia. Kokonaislukuvaatimukset nostavat kuitenkin tehtävän vaikeutta. Lisäksi myös saatavissa olevien algoritmien määrä on vähäisempi. Kokonaislukutehtävä voidaan tehdä myös jatkuvana ja hakea kokonaislukuoptimi löydetystä jatkuvasta optimista toisella menetelmällä. Tällä päästään usein riittävän hyvään tulokseen, kuten on päästy tässäkin työssä verrattuna kokonaislukutehtävänä ajettuun tehtävään.

Kvadraattisen mallin optimointi on huomattavasti vaikeampi tehtävä, sillä ne ovat käytännössä epäkonvekseja ja näihin tehtäviin sopivia algoritmeja on tällä hetkellä vähän. Vaihtoehtoisia epälineaarisiin tehtäviin sopivia algoritmeja on suositeltavaa selvitellä ja lisätä työkaluun, koska ainoa laskentatyökaluun implementoitu algoritmi on ilmainen vain akateemisen käyttöön.

Todellisten optimointitehtävien ollessa usein sekalukutehtäviä, puuttuu työkalusta vielä vaihtoehto kaupalliselle algoritmille lineaariseen sekalukuoptimointiin. Vastaavasti tarvitaan myös epälineaarisia sekalukutehtäviä tehtäviä varten vaihtoehtoinen algoritmi. Algoritmeja hakiessa ei kannata unohtaa avoimen lähdekoodin algoritmeja.

MATLAB-ohjelmassa on laaja optimointikirjasto, joista ainakin tässä työssä mainittuja epälineaarisille tehtäville tarkoitettuja algoritmeja on suositeltavaa kokeilla myös laskentatyökalussa. MATLABin geneettistä algoritmia voidaan käyttää myös epälineaarisiin sekalukutehtäviin. Sen tarkkuutta ja tehokkuutta on suositeltavaa selvittää ennen työkaluun implementointia.

Kehitetyn laskentatyökalun käyttämisen mukavuutta lisäävät tehtävien jonoon laittaminen, Techila-laskentaympäristön käyttömahdollisuus sekä ajonaikaiset hyvin informatiiviset kuvaajat sekä monipuoliset muut raportit. Nämä tallentuvat talteen tarkempaa analysointia varten. Lisäksi, aikaisempien tehtävien parametrien uudelleen hyö-

dyntämisen mahdollisuus säästää käyttäjän aikaa ja vähentää virheellisten tietojen syöttämisen mahdollisuutta.

Työssä matemaattisesti haastavinta oli laatia matemaattiset mallit diskreettejä tehtäviä varten. Ratkaisu oli loppujen lopuksi varsin yksinkertainen. Laadittujen matemaattisten mallien todettiin toimivan myös implementoituna laskentatyökaluun, joten mallien hyödyntämistä voidaan suositella myös yleisempään käyttöön. Esitetyt mallit on lisäksi helppo muodostaa esimerkiksi MATLAB-ohjelmalla, kuten tässä työssä on tehty.

Työ oli varsin laaja, koska kehittämisen aikana saatiin lukuisia ideoita, joita haluttiin työkaluun lisätä. Suppeampi työ ei olisi myöskään antanut samanlaista varmuuden tunnetta työkalun sopivuudesta vaativaan simulaatiopohjaiseen optimointiin, kuten FEM-rakennemallin optimointiin. Käyttökohteita on selvästi muitakin, kuin FEM-mallin optimointi. Kehitetty työkalu vaatii vain rajapinnan ohjelmistolle, jolla vasteet lasketaan, edellyttäen lisäksi tietysti optimoinnin kohteelta sen, että se on oikealla tavalla parametrisoitu. Tällä hetkellä työkalussa on rajapinta vain ANSYS-ohjelmaan, ja seuraava luonnollinen lisäys olisikin rajapinta Abaqus-suunnitteluohjelmaan, jota myös käytetään laajasti FEM-mallien suunnitteluun.

Työssä saadut tulokset kannustavat syventymään lisää simulaatiopohjaiseen optimointiin, sillä vaikka menetelmiä on kehitetty useita, niin niitä ei laajasti vielä yrityksissä hyödynnetä. Tutkittava aihepiiri on hyvin laaja, siksi tulisi selvittää ja määritellä tarkkaan mihin kannattaa käyttää resursseja. Metamalleista työssä saadun tiedon avulla ja laskentatyökalun hyödyntämistä ajatellen, mieleen tulee muiden metamallien implementointi työkaluun. Erittäin epälineaarisissa tehtävissä, joissa on kymmeniä suunnittelumuuttujia, tulevat tarkemmat metamallit kyseeseen. Esimerkkeinä tarkemmista metamalleista mainittakoon Kriging-menetelmä ja neuroverkot. Näillä tarkemmilla metamalleilla voidaan approksimoida huomattavasti suurempi osa suunnitteluavaruudesta kuin perinteisellä korkeintaan toisen asteen vastepintamenetelmällä. Näiden tarkempien metamallien optimointiin tulevat kyseeseen yleensä modernit heurestiset optimointimenetelmät, kuten geneettiset algoritmit ja simuloitu jäähdytys. Näihin liittyen on tehty kirjallisuuskatsauksen perusteella aika paljon tieteellistä tutkimusta. Lisäksi muita metamalleihin perustumattomia menetelmiä on suositeltavaa tutkia lisää. Yhtenä ideana on myös yhdistää useampaa menetelmää simulaatiopohjaisessa optimoinnissa.

Tämä työ liittyi laajempaan Tekes-projektiin (VTT 2012), joka jatkuu vielä 2015 vuoden syksyyn asti. Toivottavaa on, että projektin hankkeet ja siihen osallistuneiden synenergia tuottavat hyviä tuloksia ja laskennallisten menetelmien hyödyntämistä saadaan yrityksissä edistettyä. Laskentatyökalun kehittäjänä, toivon sille tietysti pitkää ikää ja leviämistä yleisempään käyttöön.

LÄHTEET

Antoniou, A. & Lu, W-S. 2007. Practical Optimization: Algorithms and Engineering Applications. Springer. 670 p.

Barton, R.R. 2009. Simulation optimization using metamodels. In: Rossetti, M.D., Hill, R.R., Johansson, B. Dunkin, A. & Ingalls, R.G. (eds.). Proceedings of the 2009 Winter Simulation Conference, Date 13-16 Dec. 2009.

Barton, R.R. & Meckesheimer, M. 2006. Metamodel-Based Simulation Optimization. In: Henderson, S.G. & Nelson, B.L. (eds.). Handbooks in Operations Research and Management Science, 13, pp. 535-570.

Bazaraa, M.S, Sherali, H.D. & Shetty, C.M. 2006. Nonlinear Programming: Theory and Algorithms. New Jersey, John Wiley & Sons, Inc. 872 p.

Branke, J., Deb, K., Miettinen, K. & Slowinski, R. 2008. Multiobjective Optimization: Interactive and Evolutionary Approaches. Springer. 470 p.

Burer, S. & Letchford, A.N. 2012. Non-convex mixed-integer nonlinear programming: A survey. Surveys in Operations Research and Management Science 17, pp. 97–106.

DAKOTA. 2014. [WWW]. [viitattu 15.3.2014]. Saatavissa: http://dakota.sandia.gov/index.html.

Fu, M.C. 2002. Optimization for Simulation: Theory vs. Practice. INFORMS Journal on Computing, 14, 3, pp. 192-215.

Griva, I., Nash, S. & Sofer, A. 2009. Linear and Nonlinear Programming. Society for Industrial Mathematics. 764 p.

Gurobi, documentation. 2014. Gurobi 5.6.

Heinonen, O. & Pajunen, S. 2011. Optimal design of stiffened plate using metamodeling techniques. J Struct Mech, 44, 3, pp. 218–230.

Hock, W. & Schittkowski, K. 1981. Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems 187. Springer.

Hsu, Y., Dong, Y. & Hsu, M. 2001. A Sequential Approximation Method Using Neural Networks for Nonlinear Discrete-Variable Optimization with Implicit Constraints. JSME International Journal Series C, 44, 1, pp. 103-112.

Huang, Z., Wang, C., J., Chen & Tian, H. 2011. Optimal design of aeroengine turbine disc based on kriging surrogate models. Computers & Structures, 89, 1-2, pp. 27-37.

Isigth. 2014. [WWW]. [viitattu X.X.2014]. Saatavissa: http://www.3ds.com/products-services/simulia/portfolio/isight-simulia-execution-engine.

Kalyanmoy, D. 2000. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering 186, 2-4, pp. 311–338.

Konnik, M. 2013. Online Constrained Receding Horizon Control for Astronomical Adaptive Optics. PhD Thesis. University of Newcastle. ID 13756. 228 p.

Lemonge, A.C., Barbosa, H.J., Borges, C.C. & Silva, F.B. 2010. Constrained Optimization Problems in Mechanical Engineering Design Using a Real-Coded Steady-State Genetic Algorithm. Mecánica Computacional XXIX, 95, pp. 9287-9303.

Linderoth, J. 2005. A simplicial branch-and-bound algorithm for solving. Mathematical Programming, 103, 2, pp. 251-282.

Loh, H.T. & Papalambros, P.Y. 1991. A Sequential Linearization Approach for Solving Mixed-Discrete Nonlinear Design Optimization Problems. Journal of Mechanical Design 113, 3, pp. 325-334.

Lorenz, T.B. 2010. Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes. SIAM-Society for Industrial and Applied Mathematics. 415 p.

Matlab, documentation. 2013. R2013b.

modeFRONTIER. 2014. [WWW]. [viitattu 15.3.2014]. Saatavissa: http://www.esteco.com/modefrontier.

Montgomery, D.C. 2003. Design and Analysis of Experiments. John Wiley & Sons, Inc. 684 p.

Myers, R.H., Montgomery, D.C. & Anderson-Cook, C.M. 2009. Response Surface Methodology: Process and Product Optimization Using Designed Experiments. John Wiley & Sons, Inc. 704 p.

Nocedal, J. & Wright, S.J. 1999. Numerical Optimization. Springer. 651 p.

OPTI. 2014. [WWW]. [viitattu X.X.2014]. Saatavissa: www.i2c2.aut.ac.nz/Wiki/OPTI.

OPTIMUS. 2014. [WWW]. [viitattu 15.3.2014]. Saatavissa: http://www.noesissolutions.com/Noesis/optimus-details/optimus-design-optimization.

Pajunen, S. & Heinonen, O. 2013. Automatic design of marine structures by using successive response surface method. Structural and Multidisciplinary Optimization, October 2013.

Ramu, M., Prabhu Raja, V., Thyla, P.R. & Gunaseelan, M. 2010. Design Optimization of Complex Structures Using Metamodels. Jordan Journal of Mechanical and Industrial Engineering, 5, 5, pp. 653-664.

Rao, S.S. 2009. Engineering Optimization: Theory and Practice. John Wiley & Sons, Inc. 840 p.

Redhe, M., Forsberg, J., Jansson, T., Marklund, P.O. & Nilsson, L. 2002. Using the response surface methodology and the D-optimality criterion in crashworthiness related problems. Structural and Multidisciplinary Optimization, 24, 3, pp. 185-194.

Roux, W.J., Stander, N. & Haftka, R.T. 1998. Response surface approximations for structural optimization. Int. J. Numer. Meth. Engng. 42, pp. 517-534.

Shan, S. & Gary Wang, G. 2010. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. Structural and Multidisciplinary Optimization, 41, 2, pp. 219–241.

Simpson, T.W., Peplinski, J.D., Koch, P.N. & Allen, J.K. 2001. Metamodels for Computer-based Engineering Design: Survey and recommendations. Engineering with Computers 17, pp. 129-150.

Stander, N. & Craig, K.J. 2002. On the robustness of a simple domain reduction scheme for simulation-based optimization. Eng. Comput., 19, 4, pp. 431-50.

Techila, documentation. 2014. Techila Fundamentals.

Thanedar, P. & Vanderplaats, G. 1995. Survey of Discrete Variable Optimization for Structural Design. Journal of Structural Engineering 121, 2, pp. 301-306.
Walpole, R.E., Myers, R.H., Myers, S.L. & Ye, K. 2007. Probability and Statistics for Engineers and Scientists. Prentice Hall. 816 p.
Wolpert, D,H. & Macready, W.G. 1997. No Free Lunch Theorems for Optimization. IEEE Transactions on evolutionary computation 1, 1, pp. 67-82.
Wolsey, L.A. 1998. Integer programming. John Wiley & Sons, Inc. 288 p.
Wright, S.J. 1997. Primal-Dual Interior-Point Methods. 309 p.
VTT. 2012. Laskennalliset menetelmät konetekniikan tuotekehityksessä. [WWW]. [viitattu 18.3.2014]. Saatavissa: http://www.vtt.fi/vtt_show_record.jsp?target=tutk&form=se&search=9944.

LIITE 1: TEHTÄVÄÄN LIITTYVÄT PARAMETRIT

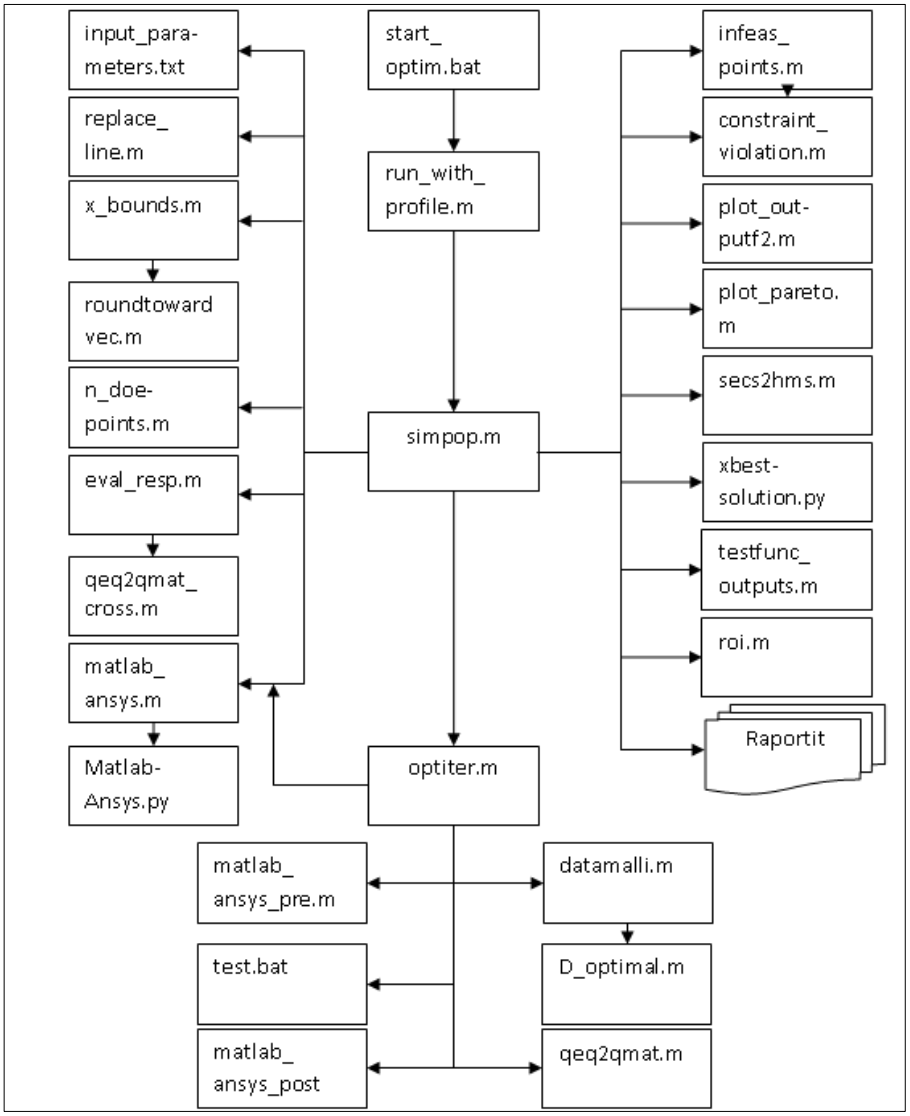
.

Parametri	Tyyppi	Kuvaus
Use Techila	kyllä / ei	Techila-laskentaympäristön käyttö
Solver and Algorithm	Valikko	Algoritmin valinta, katso luvusta 4.2 taulukko 14.
Design matrix, Model	Valikko	Malli: Linear, Quadratic, Pure Quadratic tai Lin / quad (discrete sets)
Multi-objective method	Valikko	Monitavoiteoptimoinnin menetelmä: None, Linear scalarization, Pre-emptive optimization tai Minimax
ANSYS File Location	Tiedostopolku	Optimoitavan ANSYS-mallin tiedostopolku
Test matrix	Matriisi tai MATLAB-tiedoston nimi	Testiprobleema, joka voidaan antaa matriisina tai MATLAB-tiedostona, jossa tehtävä on määritelty.
Objective functions	Vaakavektori	Kohdefunktioiden ID:t
Weights for Lin. scalarization	Vektori	Painokertoimet lineaarista skalarisointia varten.
Number of output parameters	Luku	Vasteiden lukumäärä
Target relative error	Luku	Tavoitevirhe
A, b	Vaakavektori	Lineaariset epäyhtälörajoitteet, ID:t ja ylärajat
Aeq, beq	Vaakavektori	Lineaariset yhtälörajoitteet, ID:t ja ylärajat
Qc, rhs	Vaakavektori	Kvadraattiset epäyhtälörajoitteet, ID:t ja ylärajat
TolCon	Luku	Virhetoleranssi rajoitusyhtälöille
conplot	Vektori	Kuvaajaksi tulostettavien rajoitefunktioiden ID:t
Global constraints	*nx2*-matriisi	Globaalit rajat suunnittelumuuttujille
Design variables	Vaakavektori	Suunnittelumuuttujien ID:t
Initial design points	Vektori	Aloituspiste
Variable type	Merkkijono	Muuttujien tyypit, C (jatkuva), I (kokonaisluku), B (binääri), S (diskreetti joukko) tai D (riippuva diskreetti joukko)
Precision of variables	Vektori	Suunnittelumuuttujien tarkkuus
desplot	Vektori	Kuvaajaksi tulostettavien suunnittelumuuttujien ID:t
ROI	Luku	Suunnitteluavaruuden koko, annetaan ROI:n säde
Adaptive ROI	kyllä / ei	Adaptiivinen ROI
Relative convergence tolerance	Luku	Lopetusehto, suhteellinen korvergenssi
Max iterations	Luku	Iteraatioiden maksimimäärä
RMSE	Luku	Keskineliövirhe. Ei käytössä
Discrete sets	Vaakavektorit	Diskreetit joukot
Dependencies for discrete sets	Matriisi	Riippuvien muuttujien riippuvuudet
Task Name	Merkkijono	Tehtävän nimi, ei pakollinen

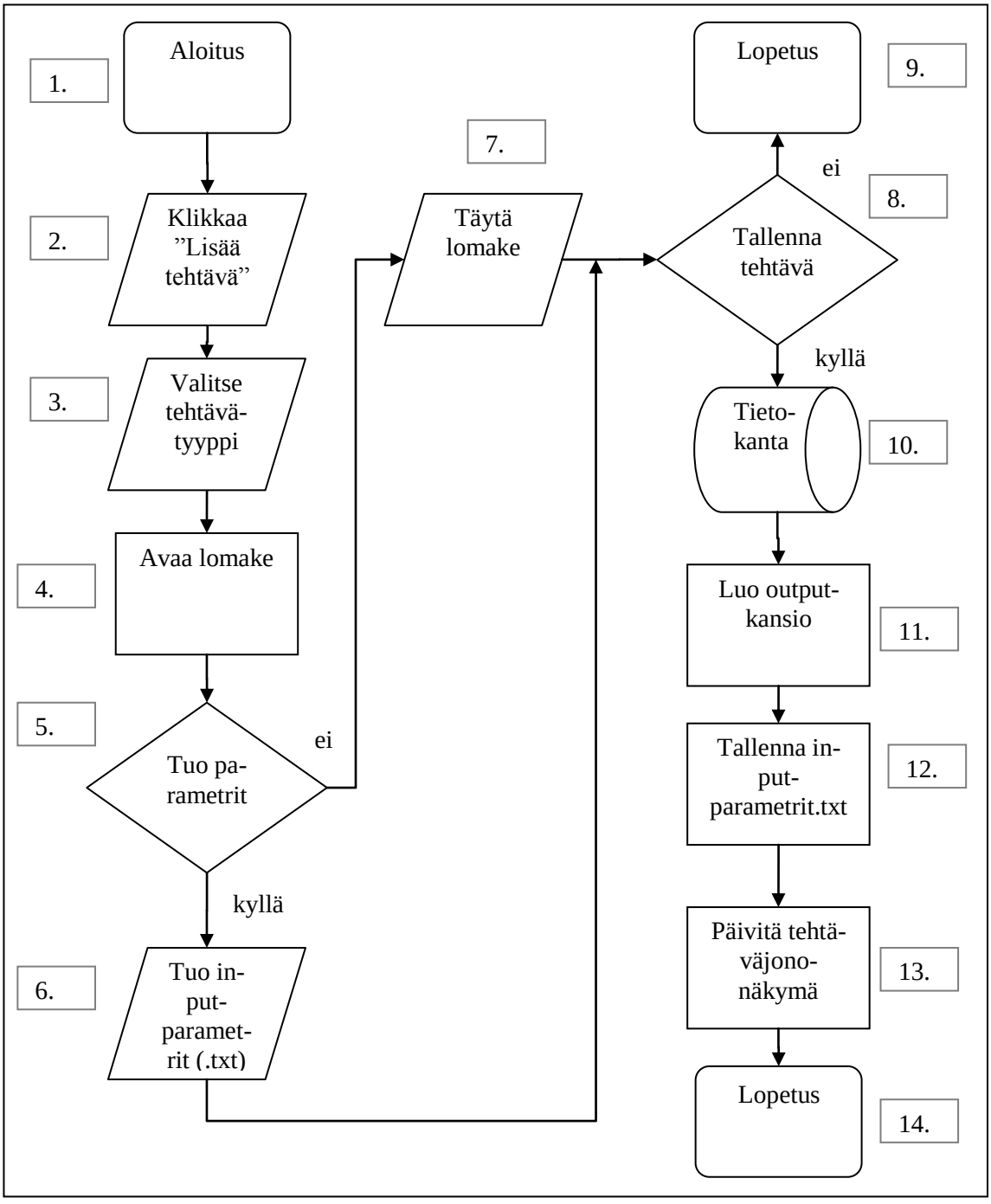
LIITE 2: LASKENTATYÖKALUN TUOTTAMAT RAPORTIT JA MUUT TIEDOSTOT

Tiedosto	Kuvaus
best_DataMatrix.txt	Parhaimman tuloksen suunnittelumuuttujat
best_solution.txt	Yhteenveto parhaimmasta tuloksesta
bounds_history.txt	Historia laskennassa käytetyistä rajoista
c_history.txt	Historia laskennassa käytetyistä ROI:n arvoista
const_xx.png	Valittujen rajoitefunktioiden kuvaajat
const_xx.txt	Historia valittujen rajoitefunktioiden arvoisya
const_xx_true.png	Valittujen rajoitefunktioiden kuvaajat
const_xx_true.txt	Historia valittujen rajoitefunktioiden arvoista
convtol_history.txt	Historia suhteellisesta konvergenssista
DataMatrix.txt	Laskennassa viimeiseksi käytetyt suunnittelumuuttujat
desplot.png	Valittujen suunnittelumuuttujien kuvaajat
des_vars.txt	Historia valittujen suunnittelumuuttujien arvoista
desvar.png	Suunnittelumuuttujien kuvaajat
dRE_history.txt	Historia käytetyistä koepisteistä
info.txt	Laskentatyökalun versio
input_parameters.txt	Tehtävässä käytetyt parametrit
iter_report.txt	Iteraatioraportti
log_file.txt	MATLABin komentoikkunan tulosteet
model_history.txt	Historia käytetyistä malleista
model_response_history.txt	Historia mallin antamista vasteista
obj_xx.png	Kohdefunktioiden kuvaajat (malli)
obj_xx.txt	Historia kohdefunktioiden arvoista (malli)
obj_xx_true.png	Kohdefunktioiden kuvaajat (todellinen vaste)
obj_xx_true.txt	Historia kohdefunktioiden arvoista (todellinen vaste)
open_best_solution.bat	Komentotiedosto parhaimman tuloksen ANSYS-ohjelmaan avaamista varten
output_history.txt	Historia todellisista vasteista
OutputValues.txt	Laskennassa viimeiseksi saatu kohdefunktion arvo
relerror_history.txt	Historia vasteiden suhteellisista virheistä
solver_history.txt	Historia käytetyistä algoritmeista
top_list.txt	Tulostiedosto, jossa tulokset lajiteltu paremmuusjärjestykseen
variables.mat	Laskentakoodin tuottamat MATLABin työpöytämuuttujat
xbestsolution.py	Aputiedosto parhaimman tuloksen ANSYS-ohjelmaan avaamista varten
z_opt_xx.png	Scipin piirtämät kuvaajat

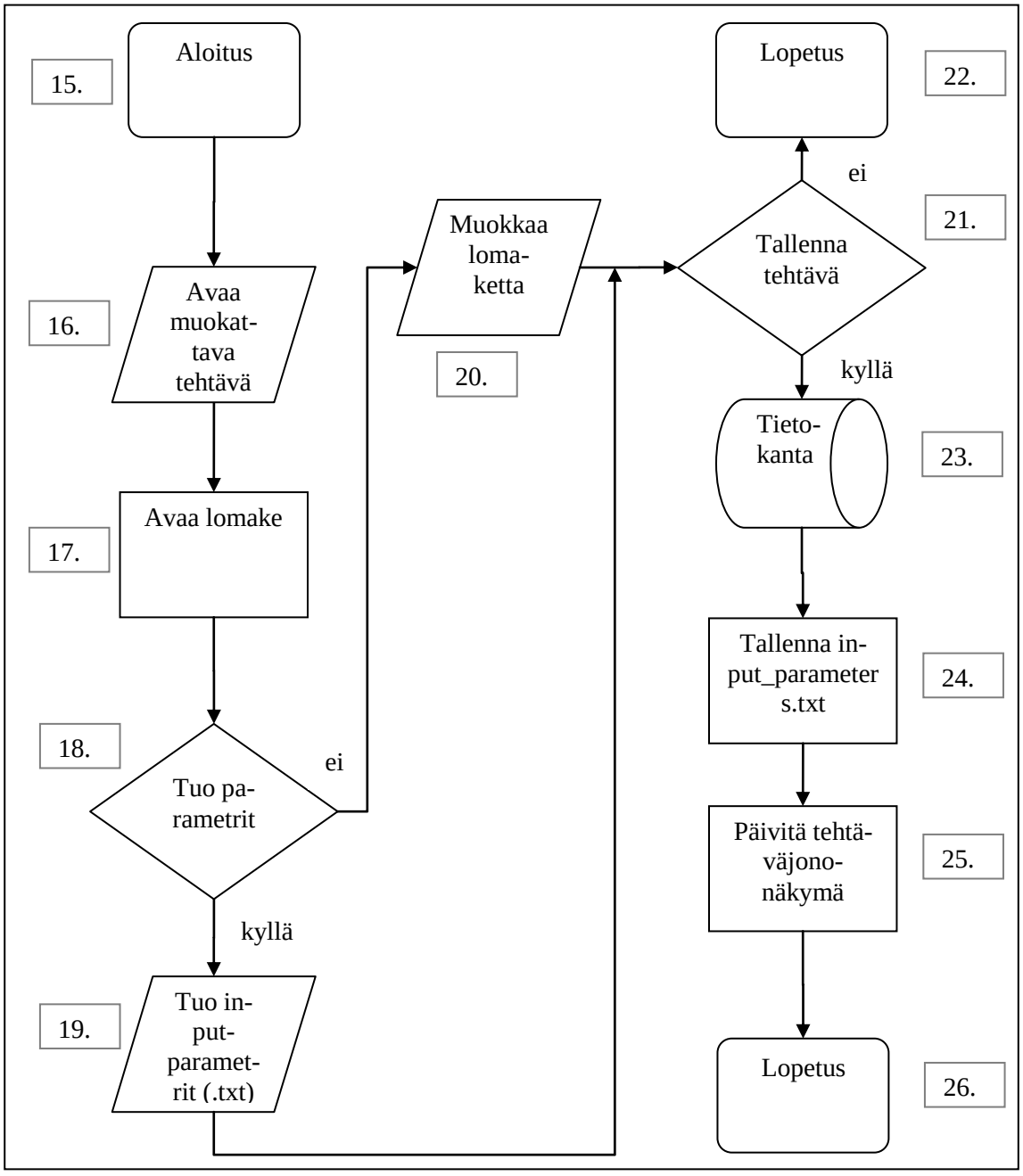
LIITE 3: LASKENTAKOODIIN LIITTYVÄT TIEDOSTOT



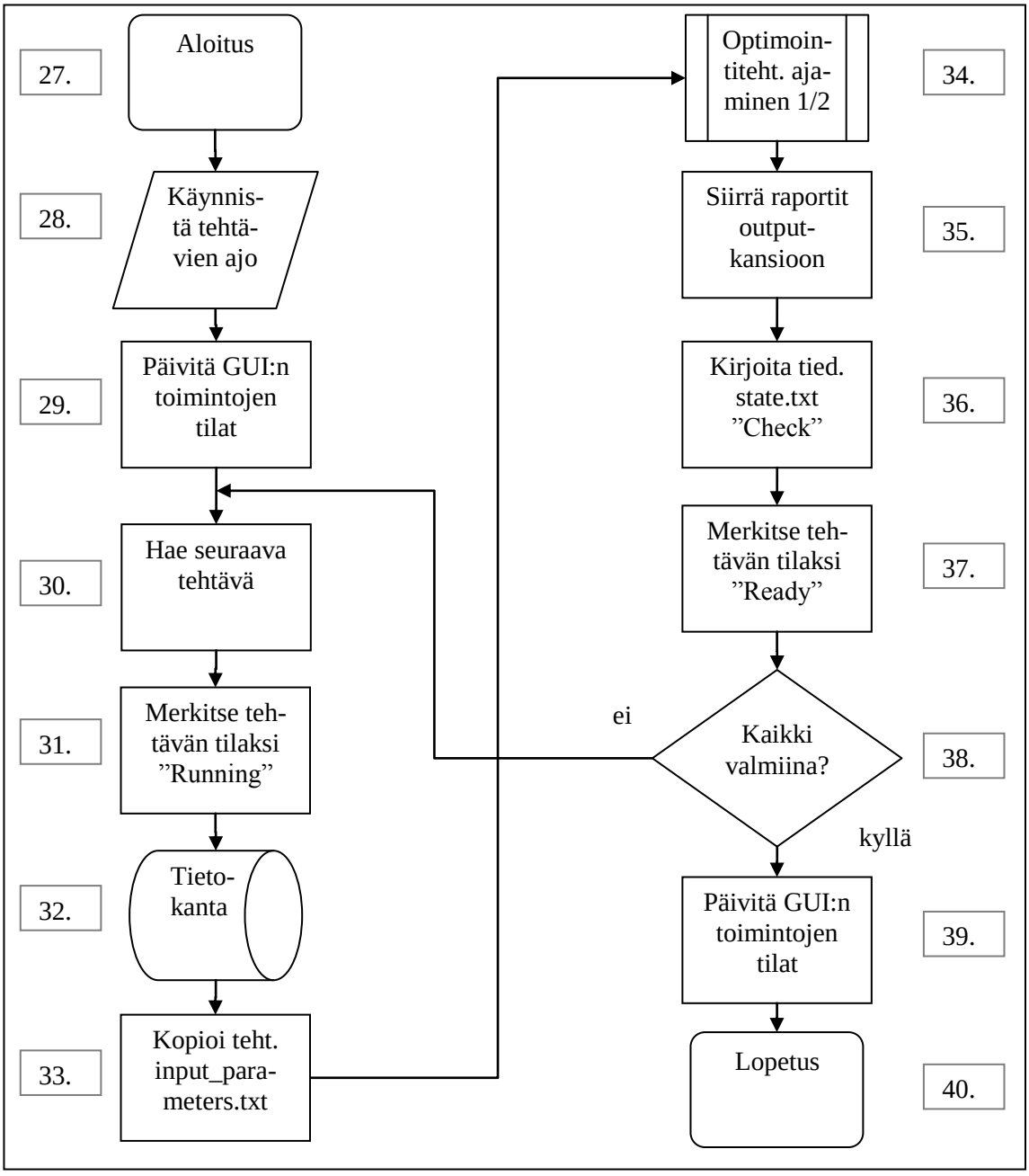
LIITE 4: TEHTÄVÄN LISÄÄMINEN



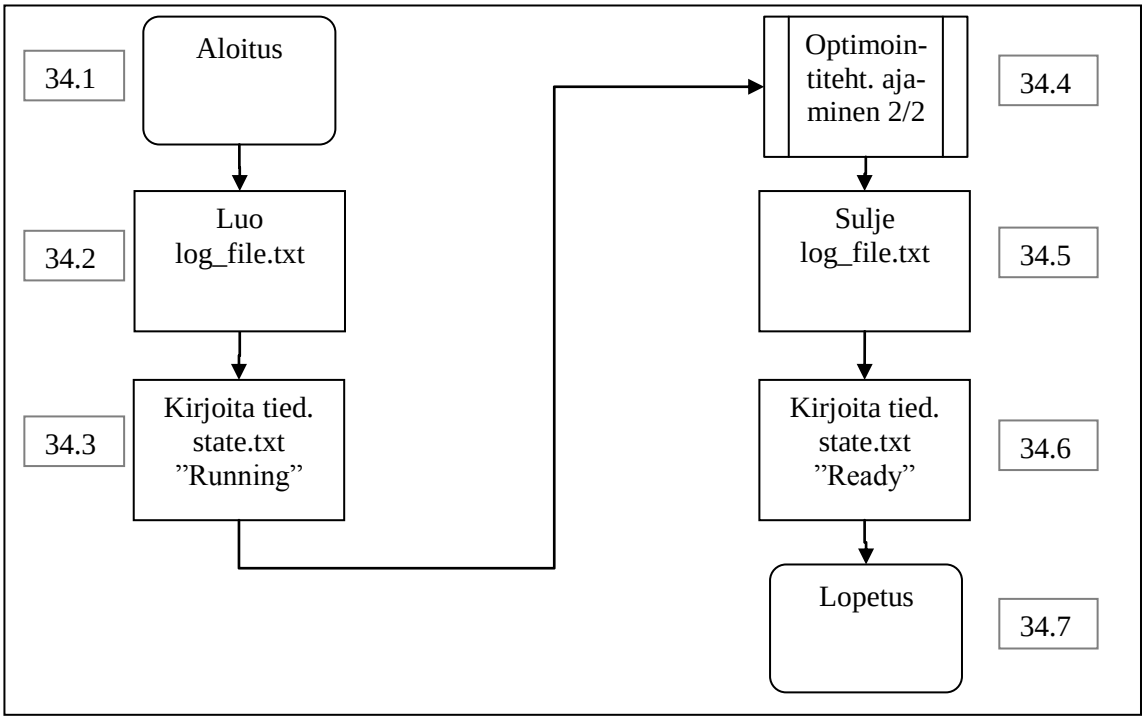
LIITE 5: TEHTÄVÄN MUOKKAAMINEN



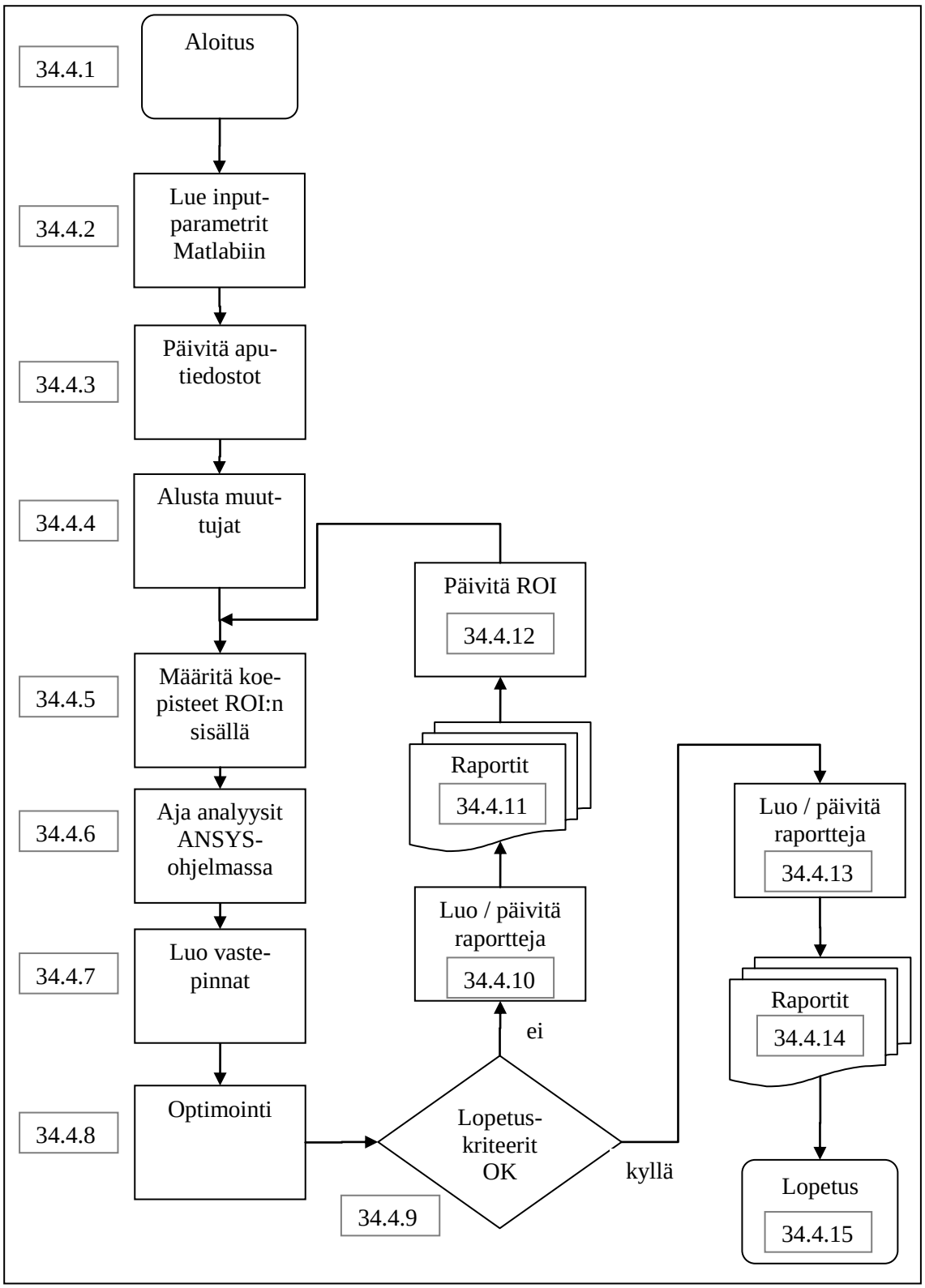
LIITE 6: TEHTÄVIEN AJAMINEN



LIITE 7: OPTIMOINTITEHTÄVÄN AJAMINEN 1/2



LIITE 8: OPTIMOINTITEHTÄVÄN AJAMINEN 2/2



LIITE 9: PARAMETRITIEDOSTO INPUT_PARAMETERS.TXT

input_parameters.txt

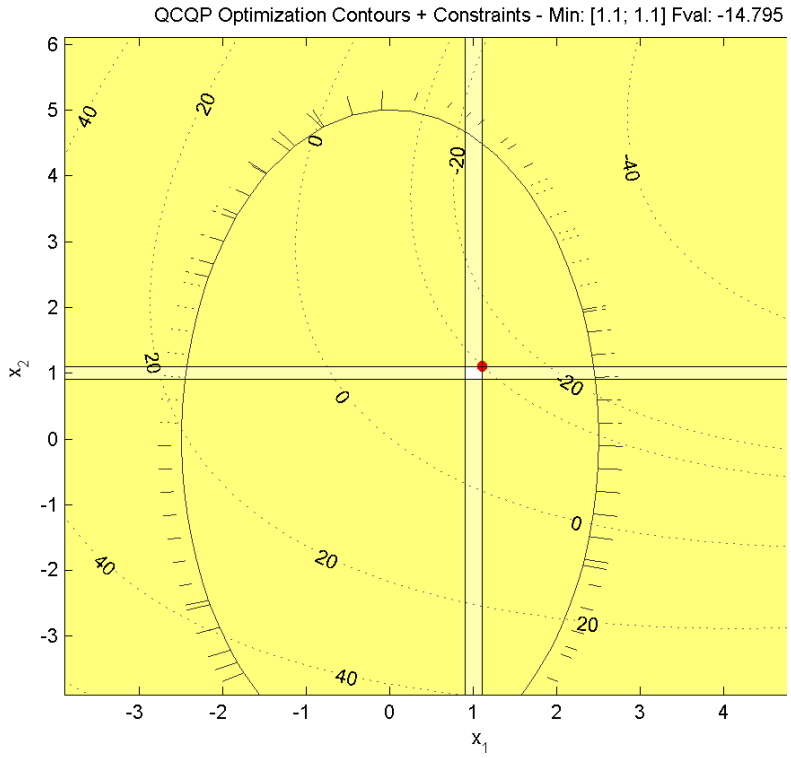
```
****solver
3.1 Discrete sets : GUROBI/SCIP
****model
lin_quad
****Ansys File Location (AnsysFileLocation)
dummy_test.wbpz
****f_id
11
****A_id
12:22
****b_values, gvec
zeros(11,1)
****Aeq_id
[ ]
****beq_values, gvec
[ ]
****Qc_id
[ ]
****Qc_rhs, gvec
[ ]
****glob_rajat (Global constraints)
[1,5;30,65;2.4,3.1;45,60;2.4,3.1;45,60;1,5;30,65;1,5;30,65]
****targetrelerror (Target relative error)
0.05
****conplot
[15;17]
****xdesvar (Initial design point)
[5;62;3.1;60;2.8;55;2.9;57;2.4;47]
****Variable type
IISDDSCCCC
****OutputParametersNum
12
****Initial hypercube ROI (c)
0.2
****Relative convergence tolerance (convtol)
1e-6
****Adaptive ROI (c)
1
****Max iterations
```

LIITE 10: SUUNNITTELUAVARUUS ITEROINTIEN AIKANA

Testiprobleema TP 12 (Hock & Schittkowski 1981):
$f(\boldsymbol{x}) = 0.5x_1^2 + x_2^2 - x_1x_2 - 7x_1 - 7x_2$
$g_1(\boldsymbol{x}) = 25 - 4x_1^2 - x_2^2 \geq 0$
Aloituspiste: $\boldsymbol{x}_0 = (1, 1), f(\boldsymbol{x}_0) = -13.5$

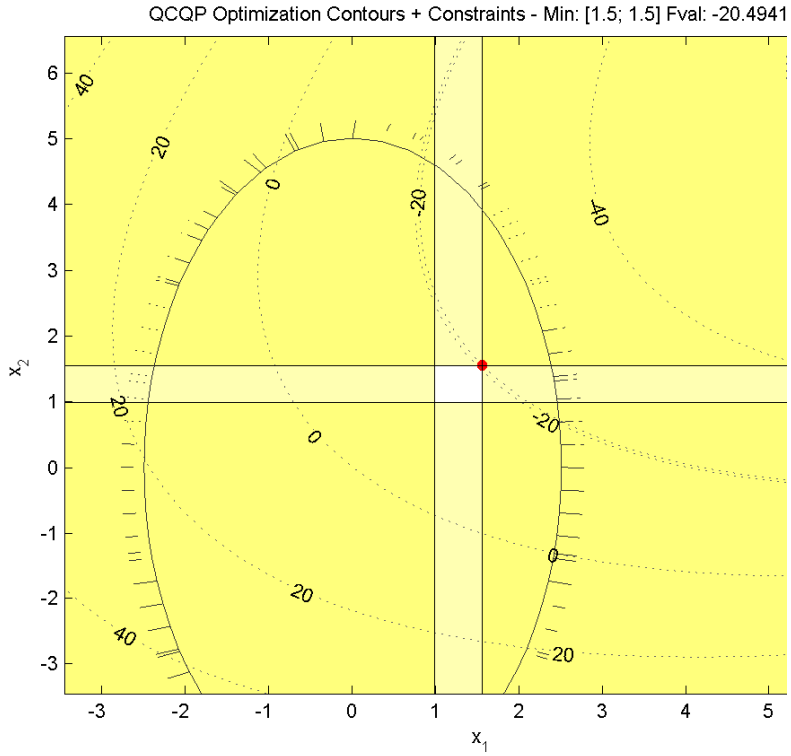
Laskentatyökalun asetukset:
Malli: Kvadraattinen, Algoritmi: Scip, Aloitus ROI (säde): 10 %, Adaptiivinen ROI, Tavoitevirhe: 1 %, Suunnittelumuuttujien tarkkuus: 1e-12, Lopetusehto: 1e-6.

1. iteraatio: ROI (säde): 10 %, $0.9 \leq x_1, x_2 \leq 1.1, \mathbf{x}^* = (1.1, 1.1), f(\mathbf{x}^*) = -14.795$



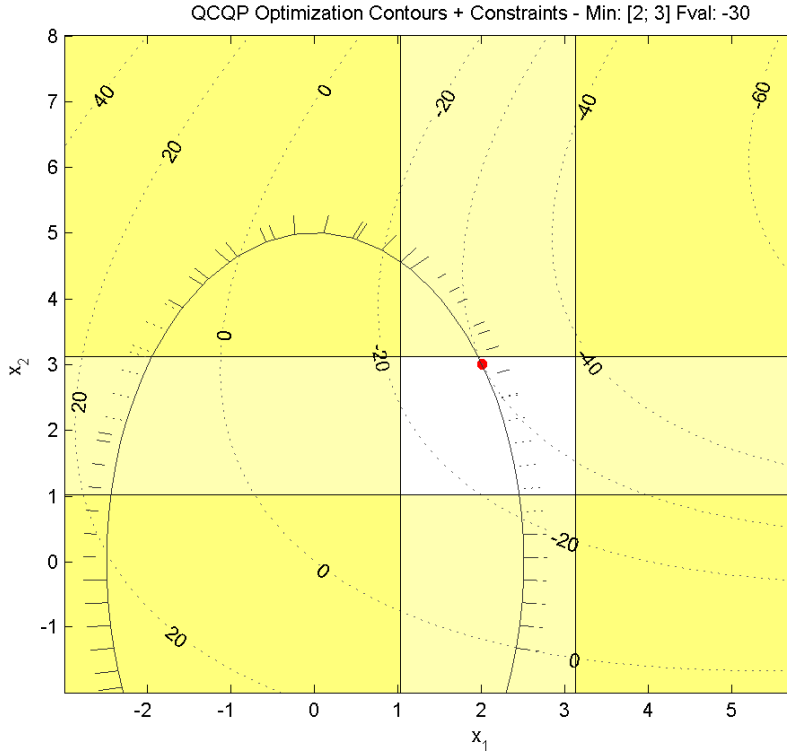
2. iteraatio: ROI (säde): 15 %, $0.935 \leq x_1, x_2 \leq 1.265$,
$\mathbf{x}^* = (1.265, 1.265), f(\mathbf{x}^*) = -16.9099$

3. iteraatio: ROI (säde): 22.5 %, $0.98 \leq x_1, x_2 \leq 1.55$,
$\mathbf{x}^* = (1.55, 1.55), f(\mathbf{x}^*) = -20.4941$



4. iteraatio: ROI (säde): 33.75 %, $1.03 \leq x_1, x_2 \leq 2.07$,
$\mathbf{x}^* = (2.07, 2.07), f(\mathbf{x}^*) = -26.8688$

5. iteraatio: ROI (säde): 50.625 %, $1.02 \leq x_1, x_2 \leq 3.12, \mathbf{x}^* = (2, 3), f(\mathbf{x}^*) = -30$



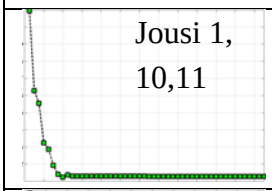
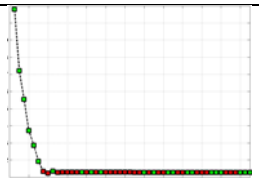
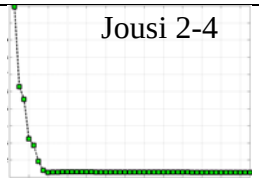
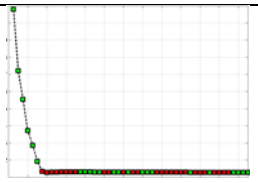
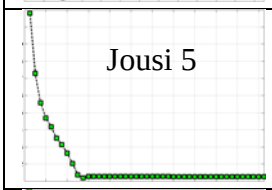
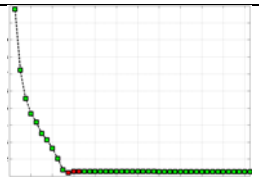
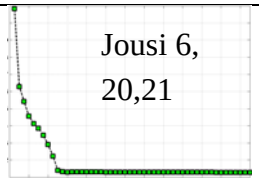
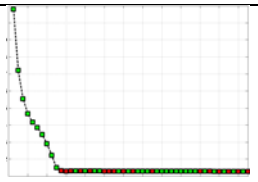
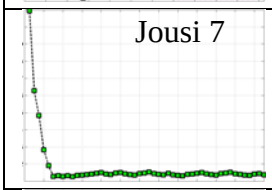
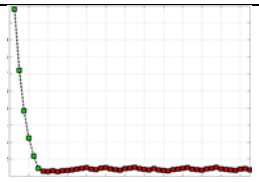
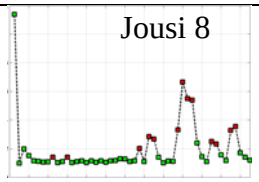
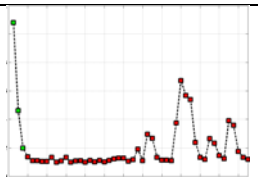
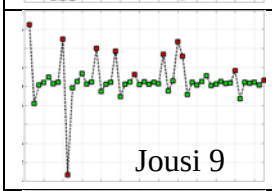
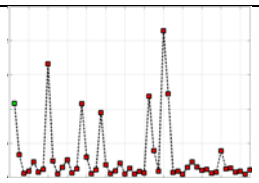
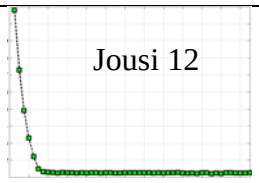
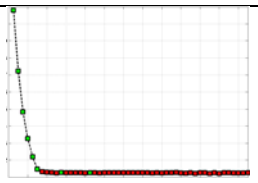
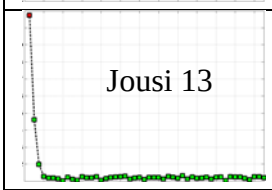
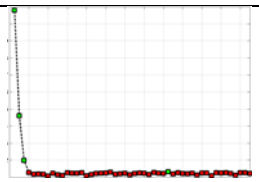
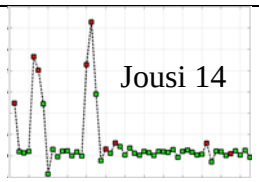
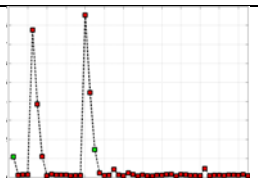
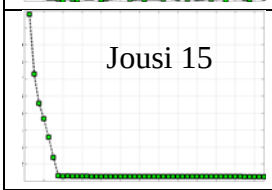
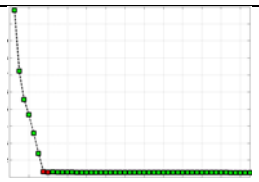
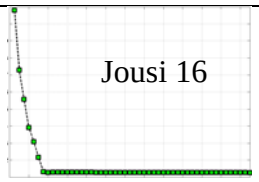
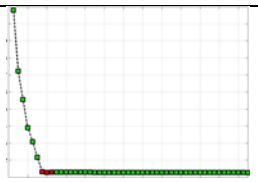
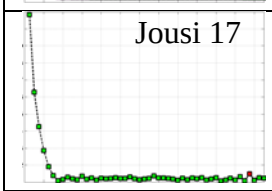
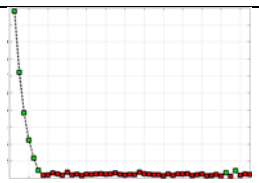
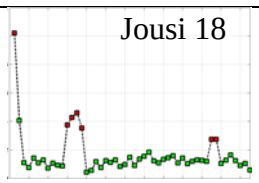
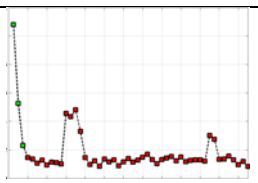
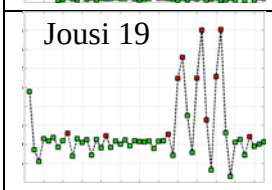
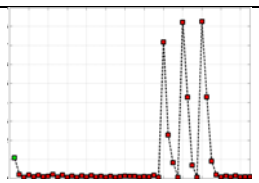
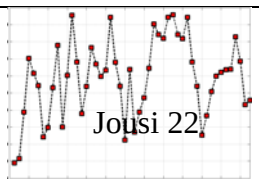
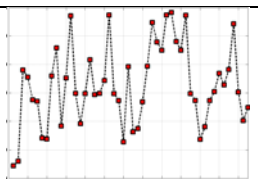
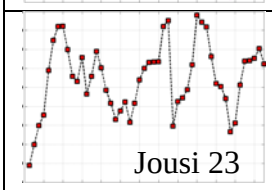
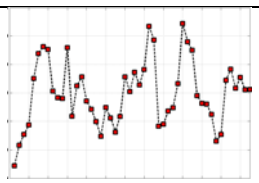
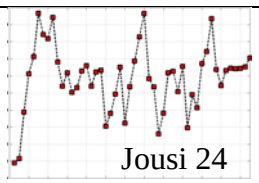
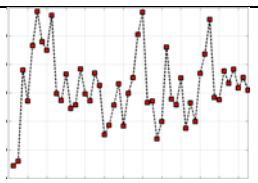
$\boldsymbol{x}^* = (2, 3)$
$f(\boldsymbol{x}^*) = -30$

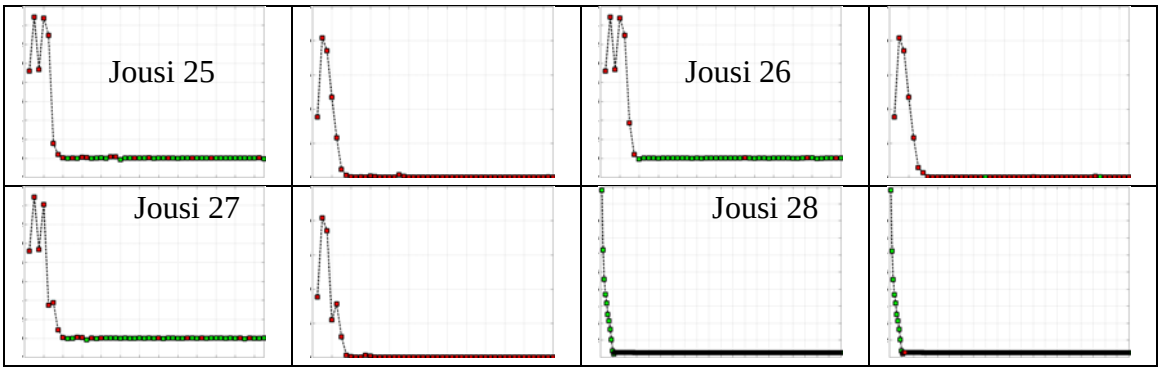
LIITE 11: LYHYT ANALYYSI KIERREJOUSEN OPTIMOINTITEHTÄVISTÄ

Tunniste	Käypien pisteiden lkm	V	Paras eikäypä (virhe)	Analyysi
Jousi 1	22/50	0.012741	0.012282 (5e-2)	Suppenee aluksi tehokkaasti, sitten hitaasti. Lähellä optimia kierroksen 8 tienoilla.
Jousi 2	23/50	0.012792	-	Suppenee aluksi tehokkaasti, sitten hitaasti. Lähellä optimia kierroksen 8 tienoilla.
Jousi 3	23/50	0.012792	-	Samat tulokset kuin Jousi 2
Jousi 4	23/50	0.012792	-	Samat tulokset kuin Jousi 2
Jousi 5	42/45	0.012676	0.012006 (3e-2)	Suppenee tehokkaasti. Lähellä optimia kierroksen 10 tienoilla. Tarkempi kuin lineaarinen.
Jousi 6	33/50	0.012831	0.012819 (1e-4)	Suppenee aluksi tehokkaasti, sitten hitaasti. Epätarkempi kuin kvadraattinen.
Jousi 7	6/50	0.014648	0.012619 (5e-2)	Suppenee tehokkaammin kiinteällä ROI:lla. Värähtelyä. Tulos on huonompi kuin adaptiivisella.
Jousi 8	3/50	0.019818	0.009842 (1e-1)	Suppenee aluksi tehokkaasti, sitten suurta värähtelyä. Liian suuri kiinteä ROI. Epätarkka.
Jousi 9	1/50	0.108000	0.004447 (2e-1)	Vain aloituspiste on käypä. Suppenee aluksi tehokkaasti, sitten todella suurta värähtelyä. Liian suuri kiinteä ROI. Todella epätarkka.
Jousi 10	22/50	0.012741	0.012282 (5e-2)	Samat tulokset kuin Jousi 1, TolRel:n kasvattamisella ei ollut vaikutusta.
Jousi 11	22/50	0.012741	0.012282 (5e-2)	Samat tulokset kuin Jousi 1, TolRel:n kasvattamisella ei ollut vaikutusta.
Jousi 12	8/20	0.012683	0.012132 (1e-2)	Suppenee tehokkaasti. Epätarkempi kiinteällä ROI:n säteellä.
Jousi 13	4/20	0.013281	0.010723 (5e-2)	Suppenee todella tehokkaasti. Pientä värähtelyä vasteissa. Vähän käypiä pisteitä.
Jousi 14	2/50	0.108000	0.005598 (2e-1)	Mallissa ja todellisessa vasteessa voimakasta värähtelyä. Interior-pointia on käytetty useassa iteraatiossa. Liian suuri kiinteä ROI. Epätarkka.
Jousi 15	48/50	0.012700	-	Suppeni tehokkaammin suuremmalla TolRel-arvolla.
Jousi 16	47/50	0.012709	-	Suppeni vielä hieman paremmin kuin Jousi 15.
Jousi 17	8/20	0.013184	0.011162 (6e-2)	Suppenee tehokkaasti, jonka jälkeen pientä värähtelyä. Interior-pointia on käytetty yhdessä iteraatiossa. Epätarkempi kiinteällä ROI:n säteellä.
Jousi 18	3/50	0.023020	0.008258 (2e-1)	Suppenee tehokkaasti, jonka jälkeen suurta värähtelyä. Interior-pointia on käytetty useassa iteraatiossa. Liian suuri kiinteä ROI. Epätarkka.
Jousi 19	1/50	0.108000	0.004161 (2e-1)	Vain aloituspiste on käypä. Interior-pointia on käytetty useassa iteraatiossa. Todella suurta värähtelyä ja on epätarkka.

Jousi 20	33/50	0.012831	0.012819 (1e-4)	Samat tulokset kuin Jousi 6, TolRel:n kasvattamisella ei ollut vaikutusta, koska virhettä on melko paljon.
Jousi 21	33/50	0.012831	0.012819 (1e-4)	Samat tulokset kuin Jousi 6, TolRel:n kasvattamisella ei ollut vaikutusta.
Jousi 22	0/50	8.800000	8.800000 (3e-1)	Todella suurta värähtelyä, ei löydä käypää aluetta. Liian pieni ROI.
Jousi 23	0/50	8.800000	8.800000 (3e-1)	Suurta värähtelyä, ei löydä käypää aluetta. Liian pieni ROI. Interior-pointia on käytetty kaikissa iteraatiossa.
Jousi 24	0/50	8.800000	8.800000 (3e-1)	Todella suurta värähtelyä, ei löydä käypää aluetta. Liian pieni ROI. Interior-pointia on käytetty kaikissa iteraatiossa.
Jousi 25	0/50	0.002675	0.002675 (2e-1)	Kasvaa ensin, mutta kääntyy laskuun ja suppenee sitten nopeasti. Ei löydä käypää aluetta. Epätarkka. Epäkäyvästä aloituksesta ei löydetä käypää pistettä.
Jousi 26	2/50	0.013051	0.004941 (2e-1)	Kasvaa ensin, mutta kääntyy laskuun ja suppenee sitten nopeasti. Interior-pointia on käytetty useassa iteraatiossa. Käypiä pisteitä löytyi 2. Alussa isompaa värähtelyä, lopuksi pienempää. Epätarkka. Löydetystä käyvästä pisteestä voitaisiin halutessa jatkaa esim. adaptiivisella menetelmällä. Tulos parani huomattavasti lähtöpisteestä.
Jousi 27	0/50	0.003769	0.003769 (2e-1)	Kasvaa ensin, mutta kääntyy laskuun ja suppenee sitten nopeasti. Interior-pointia on käytetty useassa iteraatiossa. Alussa isompaa värähtelyä, lopuksi pienempää. Ei löydä käypää aluetta. Epätarkka. Epäkäyvästä aloituksesta ei löydetä käypää pistettä.
Jousi 28	197/ 200	0.012664	0.012006 (3e-2)	Iteraatioita kasvattamalla löytyi todella lähelle kirjallisuudesta saatua optimia.

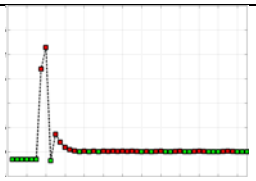
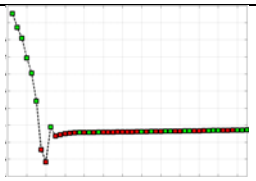
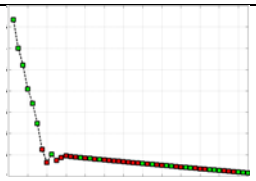
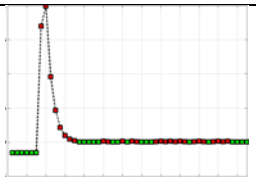
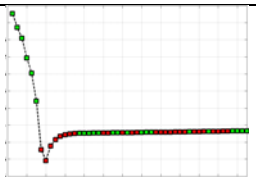
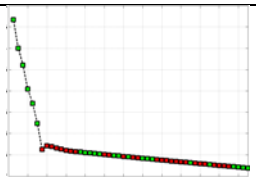
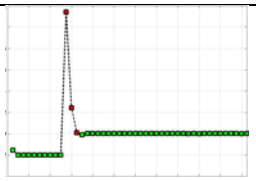
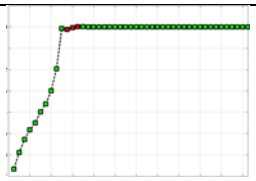
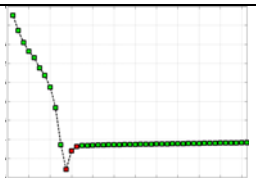
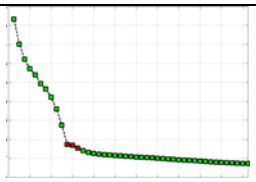
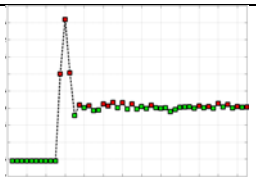
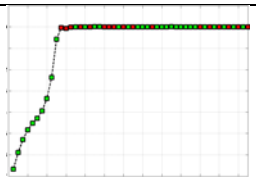
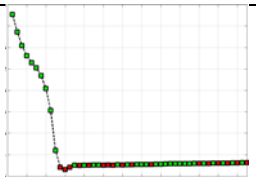
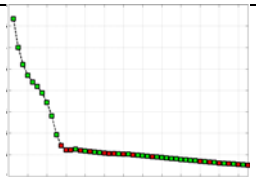
LIITE 12: KOHDEFUNKTION SUPPENEMISEN VERTAILU TEHTÄVISSÄ JOUSI 1-28

Malli	Tarkka	Malli	Tarkka
			
			
			
			
			
			
			
			
			

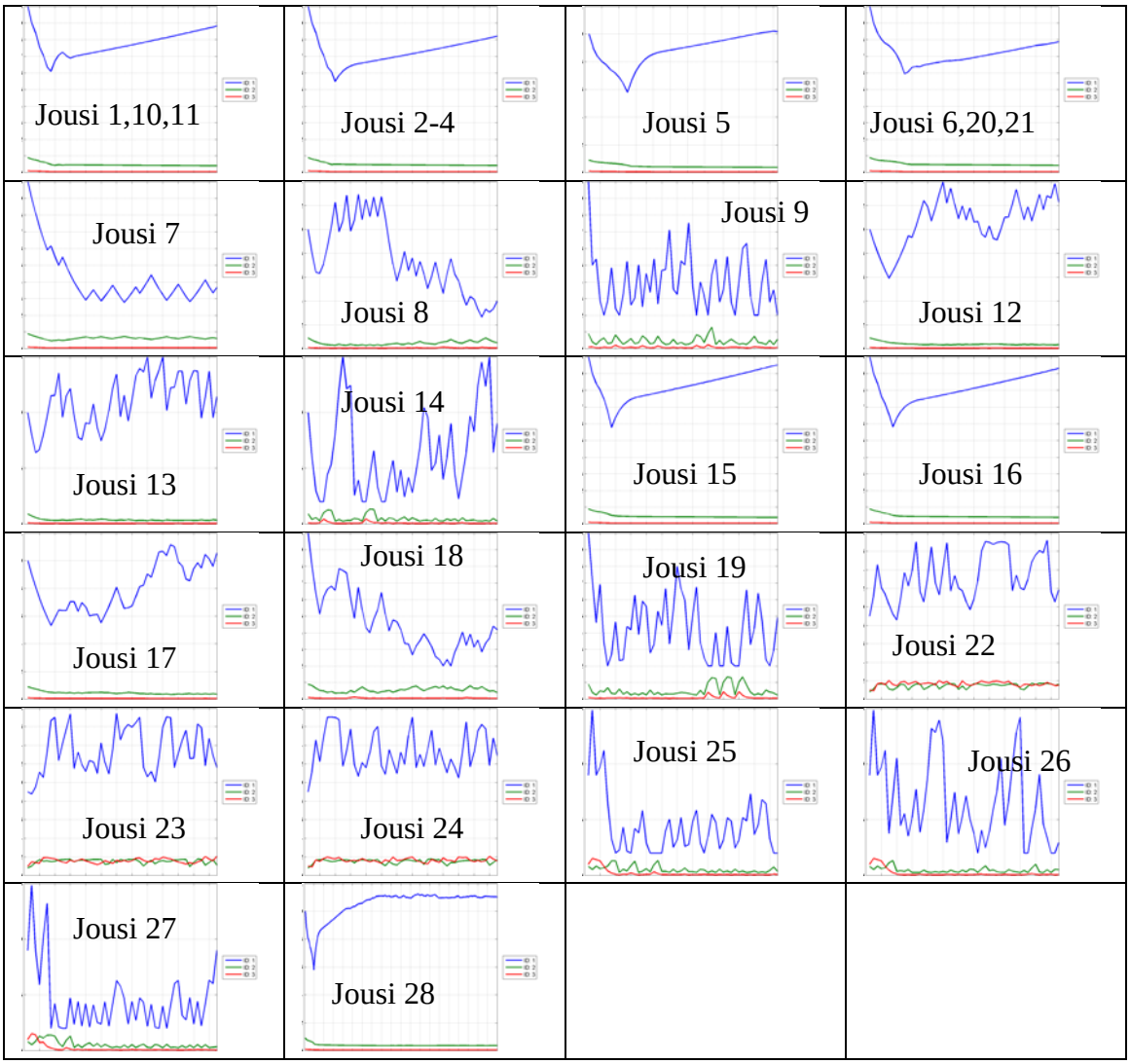

Jousi 25
Jousi 26
Jousi 27
Jousi 28

LIITE 13: RAJOITEFUNKTIOIDEN SUPPENEMISEN VERTAILU TEHTÄVISSÄ JOUSI 1-6

Verrataan lineaaristen ja kvadraattisten mallien rajoitefunktioiden suppenemista.

Malli	Tarkka	Malli	Tarkka
Jousi 1: g_1		Jousi 1: g_2	
Jousi 1: g_3		Jousi 1: g_4	
Jousi 2-4: g_1		Jousi 2-4: g_2	
Jousi 2-4: g_3		Jousi 2-4: g_4	
Jousi 5: g_1		Jousi 5: g_2	
Jousi 5: g_3		Jousi 5: g_4	
Jousi 6: g_1		Jousi 6: g_2	
Jousi 6: g_3		Jousi 6: g_4	

LIITE 14: MUUTTUJIEN SUPPENEMISEN VERTAILU TEHTÄVISSÄ JOUSI 1-28



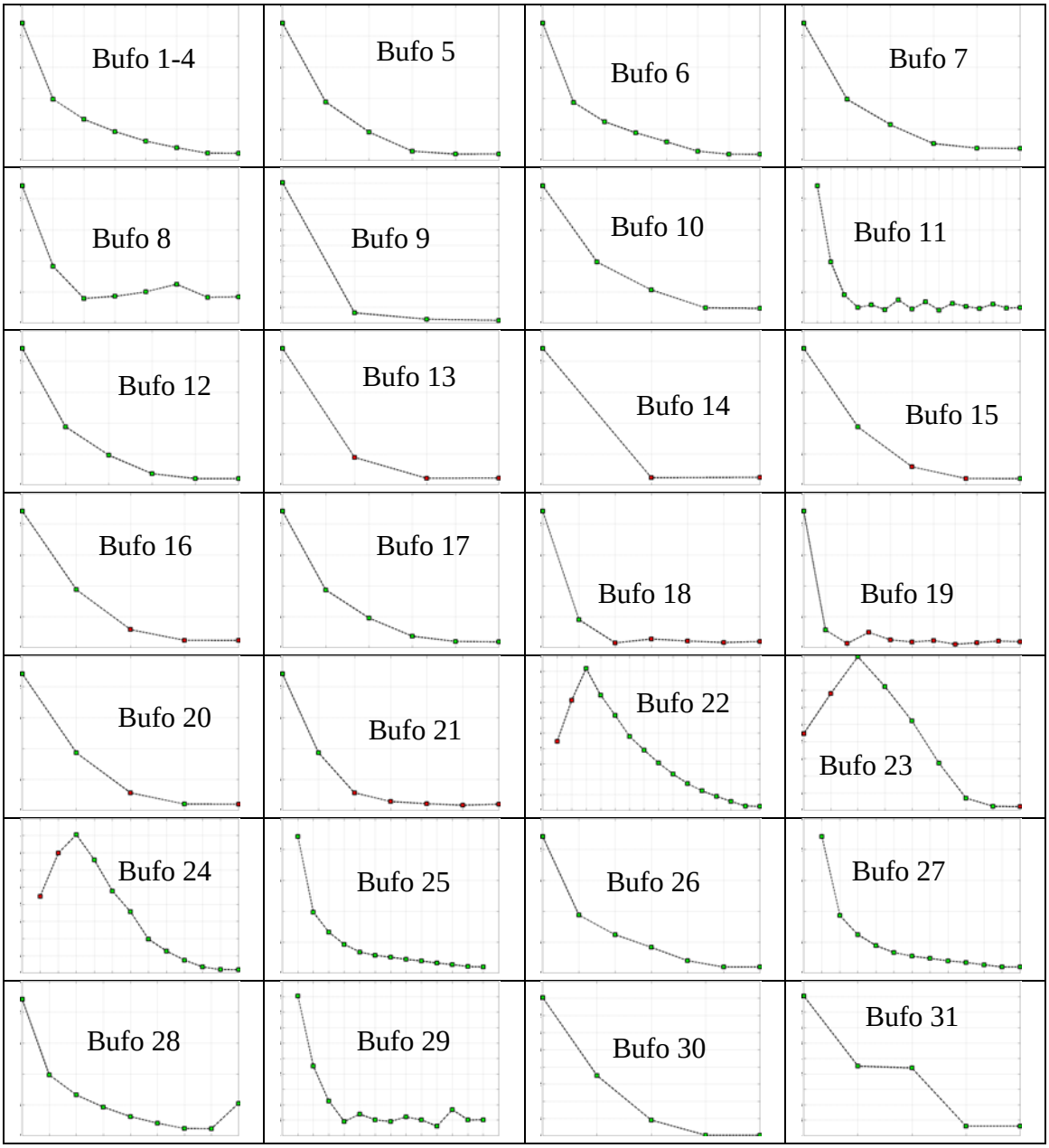
LIITE 15: LYHYT ANALYYSI LASTILUUKUN OPTIMOINTITEHTÄVISTÄ

Tunniste	Käypien pisteiden lkm	m	Paras ei-käypä (virhe)	Analyysi
Bufo 1	8/8	20112	-	Suppenee tehokkaasti, vaikka ROI:n säde pienenee
Bufo 2	8/8	20112	-	Suppenee tehokkaasti, vaikka ROI:n säde pienenee
Bufo 3	8/8	20112	-	Suppenee tehokkaasti, vaikka ROI:n säde pienenee
Bufo 4	8/8	20112	-	Suppenee tehokkaasti, vaikka ROI:n säde pienenee
Bufo 5	6/6	20098	-	Suppenee tehokkaasti, ROI:n säde kasvaa hieman muutamalla kierroksella
Bufo 6	8/8	20094	-	Suppenee tehokkaasti, vaikka ROI:n säde pienenee
Bufo 7	6/6	20190	-	Suppenee tehokkaammin kiinteällä ROI:n säteellä. Tulos huononee hieman.
Bufo 8	8/8	20394	-	Suppenee tehokkaasti suuremmalla kiinteällä ROI:n säteellä. Tulos huononee selvästi. Kaikki pisteet täysin käypiä vaikka ROI:n säde on suuri.
Bufo 9	4/4	20432	-	Suppenee todella tehokkaasti kiinteällä ROI:n säteellä (50 %). Tulos huononee selvästi. Kaikki pisteet täysin käypiä vaikka ROI:n säde on suuri.
Bufo 10	5/5	20234	-	Suuremmalla TolRel suppenee tehokkaammin, ROI:n säde kasvaa hieman muutamalla kierroksella. Tulos huononee hieman. Tarkka, vain viimeisellä kierroksella virhettä (TolCon).
Bufo 11	16/16	20207	-	Suppenee tehokkaasti kun TolRel on 10 %. Vaste alkaa värähtelemään 4 kierroksen jälkeen.
Bufo 12	6/6	20098	-	Vastaava tulos kuin Bufo 5. Kesto 20 min enemmän vaikka iteraatioiden lkm on sama.
Bufo 13	1/4	22209	20105 (3e-3)	Vain aloituspiste on käypä. Kahdella iteraatiolla lähelle käypää optimia.
Bufo 14	1/3	22209	20113 (2e-3)	Vain aloituspiste on käypä. Yhdellä iteraatiolla lähelle käypää optimia. Kvadraattinen toimii tässä hyvin isollakin ROI:lla.
Bufo 15	3/5	20100	-	Suppenee nopeammin kuin Bufo 5. Kesto yli 2,5 h. Vähän laskentayksiköitä ollut käytössä.
Bufo 16	2/5	20938	20114 (3e-3)	Melkein sama kuin Bufo 15 ajo. Epätarkempi.
Bufo 17	6/6	20091	-	Suppenee tehokkaammin kiinteällä ROI:lla. Paras tulos.
Bufo 18	2/7	20449	20074 (4e-3)	Suppenee tehokkaasti, jonka jälkeen pientä värähtelyä. Epätarkempi.
Bufo 19	2/11	20282	20052 (6e-3)	Kahdella iteraatiolla lähelle käypää optimia. Suppenee tehokkaasti, jonka jälkeen värähtelyä. Epätarkempi.
Bufo 20	2/5	20099	20097 (1e-3)	Suuremmalla TolRel suppenee tehokkaammin, ROI:n säde kasvaa muutamalla kierroksella. Epätarkempi.
Bufo 21	2/7	20930	20071	Suuremmalla TolRel suppenee tehokkaammin, ROI:n säde

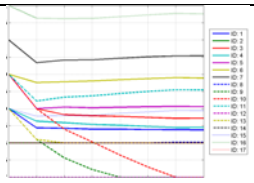
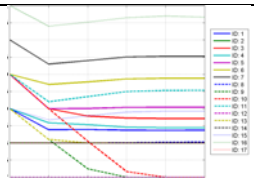
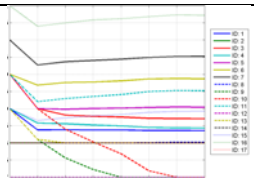
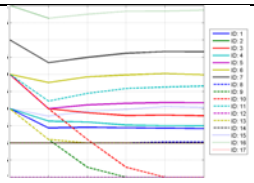
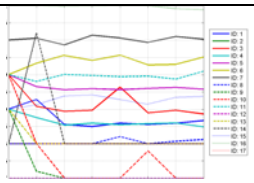
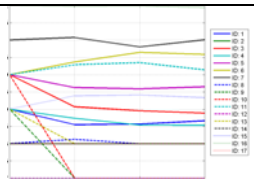
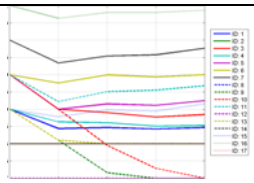
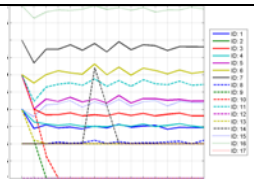
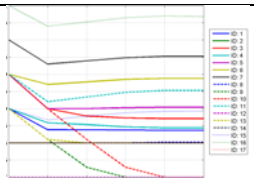
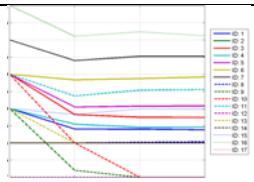
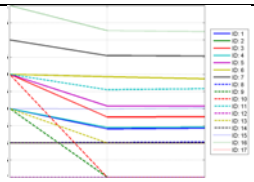
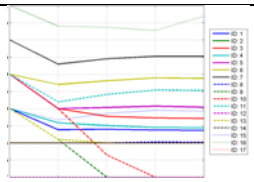
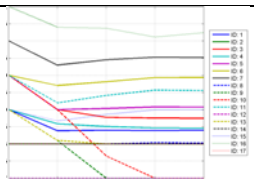
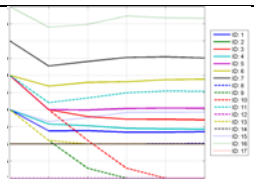
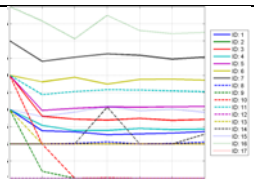
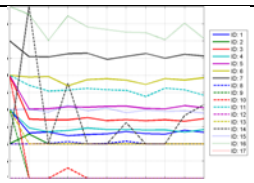
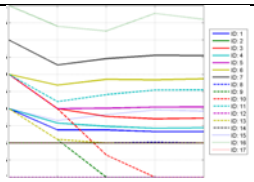
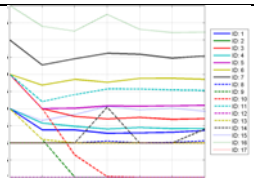
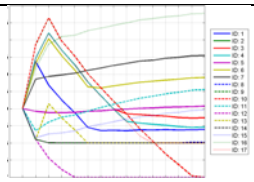
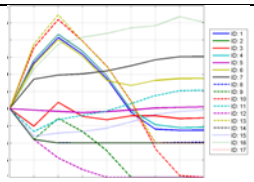
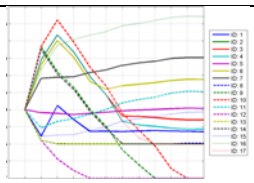
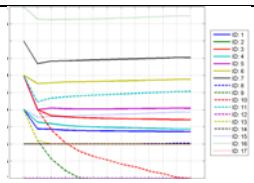
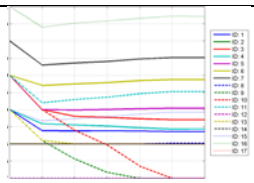
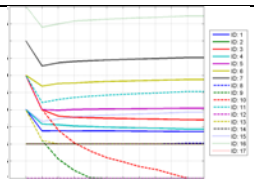
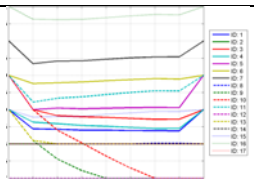
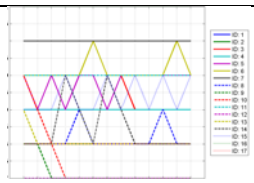
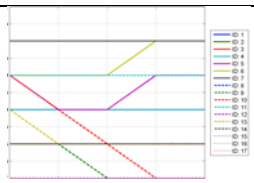
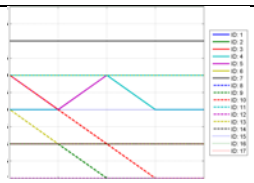
			(4e-3)	kasvaa melkein joka kierroksella. Epätarkempi.
Bufo 22	13/15	20122	-	Epäkäyvästä aloituksesta päästään lähelle Bufo 1 tulosta. Käypä piste löytyi jo ensimmäisellä (kierros 2) iteraatiolla.
Bufo 23	6/9	20110	20103 (1e-3)	Interior pointia on käytetty vain ensimmäisessä iteraatiossa (kierros 2). Epäkäyvästä aloituksesta päästään melkein samaan tulokseen kuin Bufo 5.
Bufo 24	10/12	20094	-	Interior pointia on käytetty vain ensimmäisessä iteraatiossa (kierros 2). Epäkäyvästä aloituksesta päästään samaan tulokseen kuin Bufo 6.
Bufo 25	13/13	20096	-	Vaatii 5 iteraatiota enemmän iteraatioita kuin Bufo 1. Suppenee hitaasti. Tulos parani hieman.
Bufo 26	7/7	20095	-	Suppenee hitaammin kuin Bufo 5. Hieman tarkempi.
Bufo 27	12/12	20094	-	Suppenee hitaammin kuin Bufo 6. Hieman tarkempi.
Bufo 28	9/9	20521	-	8 iteraation jälkeen diskreetin pisteen haku löytää käyvän diskreetin pisteen. Jatkuva optimi on 20112.
Bufo 29	13/13	20521	-	Lähellä kokonaislukuoptimia jo 4 kierroksen jälkeen.
Bufo 30	5/5	20602	-	Löytää jo 4 kierroksen jälkeen täysin käyvän ratkaisun. Jokaisen iteraation ratkaisu on täysin käypä. Hitain kokonaislukutehtävistä.
Bufo 31	4/5	20521	-	Löytää jo 4 kierroksen jälkeen käyvän ratkaisun. Nopein kokonaislukutehtävistä.

LIITE 16: KOHDEFUNKTION SUPPENEMISEN VERTAILU TEHTÄVISSÄ BUFO 1-31

Vertailtu vain todellisia vasteita, koska melkein kaikissa ajoissa malli approksimoi vasteita hyvin tarkasti.



LIITE 17: MUUTTUJIEN SUPPENEMISEN VERTAILU TEHTÄVISSÄ BUFO 1-31

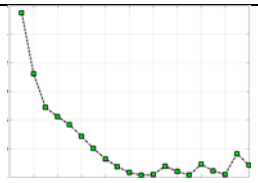
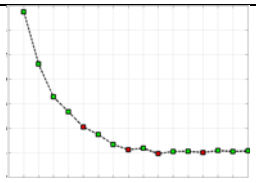
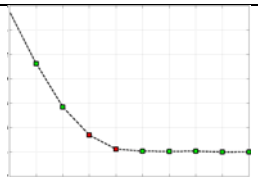
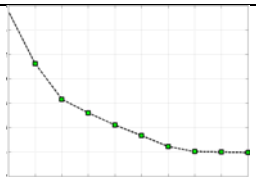
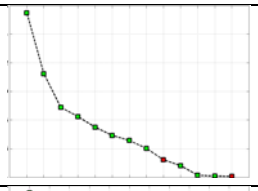
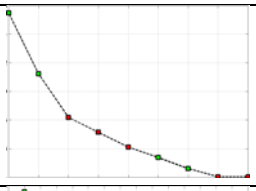
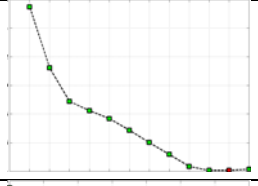
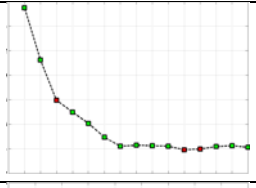
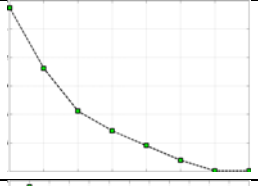
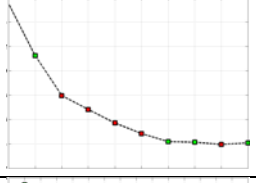
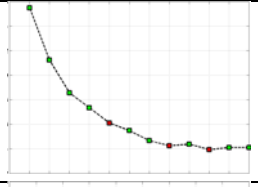
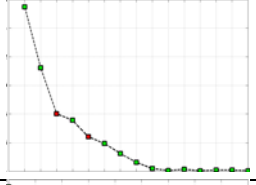
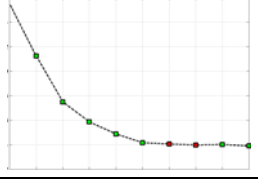
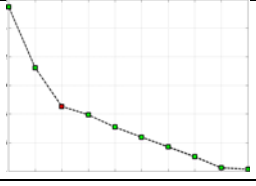
			
Bufo 1-4	Bufo 5	Bufo 6	Bufo 7
			
Bufo 8	Bufo 9	Bufo 10	Bufo 11
			
Bufo 12	Bufo 13	Bufo 14	Bufo 15
			
Bufo 16	Bufo 17	Bufo 18	Bufo 19
			
Bufo 20	Bufo 21	Bufo 22	Bufo 23
			
Bufo 24	Bufo 25	Bufo 26	Bufo 27
			
Bufo 28	Bufo 29	Bufo 30	Bufo 31

LIITE 18: LYHYT ANALYYSI DYNAAMISEN PALKIN OPTIMOINTITEHTÄVISTÄ

Tunniste	Käyp. pist. lkm	Massa	Paras ei-käypä (virhe)	Analyysi
Palkki 1	20/20	14.141742	-	Optimi löytyi 11 kierroksen jälkeen, jonka jälkeen melko voimakasta värähtelyä.
Palkki 2	12/16	14.091483	13.924180 (4e-2)	Lähellä optimia 10 kierroksen jälkeen. Ei värähtelyä kohdefunktion vasteessa.
Palkki 3	8/10	13.980928	-	Suppenee tehokkaasti, lähellä optimia jo 6 kierroksen jälkeen.
Palkki 4	10/10	13.939998	-	Suppenee tehokkaasti, lähellä optimia 8 kierroksen jälkeen.
Palkki 5	11/13	14.077802	14.066678 (1e-2)	Suppenee hitaasti. Ajettu uudestaan asettamalla max iteraatioiksi 20.
Palkki 6	4/9	14.599067	14.022778 (2e-2)	Suppenee tehokkaasti, mutta on epätarkka. Vain 4 käypää pistettä.
Palkki 7	11/12	14.137428	-	Max iteraatiot rajattiin 11 eli pakotettiin lopettamaan tähän ajan säästämiseksi. 11 iteraation jälkeen diskreetin pisteen haku löytää käyvän diskreetin pisteen. Jatkuva optimi on 14.057570, joka on epäkäypä ja on eri kuin Palkki 1 kierroksella 11saatu 14.141742. Raportteja tarkastelemalla huomataan, että ANSYS-ohjelman palauttamissa vasteissa on hyvin pientä eroa viimeisten desimaalien kohdalla. Ero vaikuttaa suhteellisten virheiden kautta adaptiiviseen ROI:n säteeseen ja sitä kautta ROI:n kokoon. Lopputulos on kuitenkin hyvä.
Palkki 8	12/15	14.108922	13.909236 (2e-2)	Suppenee tehokkaasti kiinteällä ROI:n säteellä. Lähellä optimia jo 7 kierroksen jälkeen.
Palkki 9	8/8	14.0206488	-	Suppenee tehokkaasti kiinteällä ROI:n säteellä. Kaikki pisteet käypiä, joista 5 on täysin käypää.
Palkki 10	5/10	14.075106	13.948147 (4e-2)	Suppenee tehokkaasti kiinteällä ROI:n säteellä. Epätarkempi, vain 5 käypää pistettä, jotka ovat tosin kaikki täysin käypiä.
Palkki 11	9/12	14.096157	-	11 iteraation jälkeen diskreetin pisteen haku löytää käyvän diskreetin pisteen. Jatkuva optimi on 14.102350, joka on sama kuin Palkki 2 saatu kierroksella 11. Max iteraatiot rajattiin 11 kierrokseen ajan säästämiseksi. Palkki 2 ajossa kierroksella 11 oltiin jo lähellä optimia ja piste oli täysin käypä.
Palkki 12	13/15	14.029738	-	Lähellä optimia 10 kierroksen jälkeen. Suppenee hitaammin adaptiivisella ROI:n säteellä.
Palkki 13	8/10	13.904251	-	Lähellä optimia jo 6 kierroksen jälkeen. Suppenee

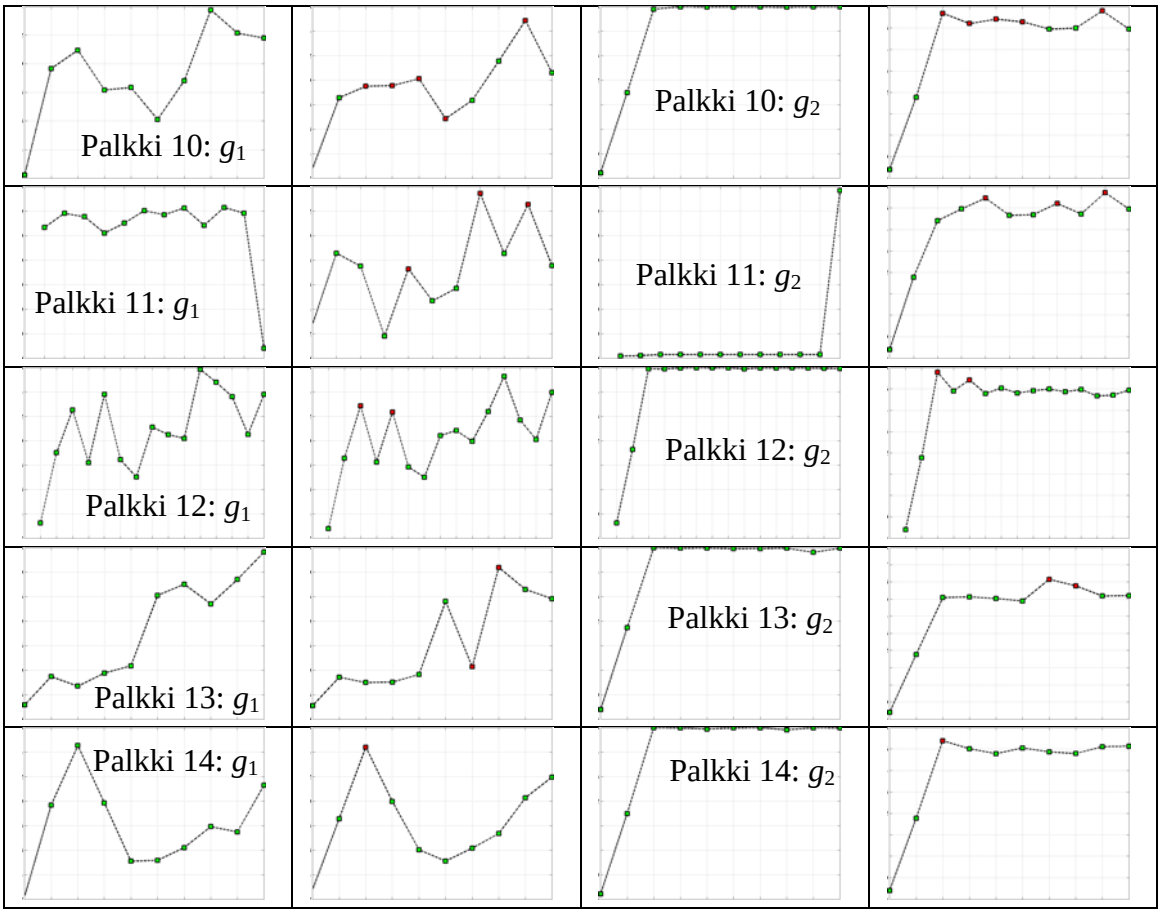
				tehokkaammin adaptiivisella ROI:n säteellä. Alussa kasvattaa ROI:n sädettä.
Palkki 14	9/10	14.143453	-	Tarkempi kuin kiinteällä ROI:n säteellä ja pääs-tään parempaan tulokseen 10 kierroksen aikana.

LIITE 19: KOHDEFUNKTION SUPPENEMISEN VERTAILU TEHTÄVISSÄ PALKKI 1-14

Malli	Tarkka	Malli	Tarkka
Palkki 1		Palkki 2	
Palkki 3		Palkki 4	
Palkki 5		Palkki 6	
Palkki 7		Palkki 8	
Palkki 9		Palkki 10	
Palkki 11		Palkki 12	
Palkki 13		Palkki 14	

LIITE 20: RAJOITEFUNKTIOIDEN SUPPENEMISEN VERTAILU TEHTÄVISSÄ PALKKI 1-14

Malli	Tarkka	Malli	Tarkka
Palkki 1: g_1		Palkki 1: g_2	
Palkki 2: g_1		Palkki 2: g_2	
Palkki 3: g_1		Palkki 3: g_2	
Palkki 4: g_1		Palkki 4: g_2	
Palkki 5: g_1		Palkki 5: g_2	
Palkki 6: g_1		Palkki 6: g_2	
Palkki 7: g_1		Palkki 7: g_2	
Palkki 8: g_1		Palkki 8: g_2	
Palkki 9: g_1		Palkki 9: g_2	


Palkki 10: g_1
Palkki 10: g_2
Palkki 11: g_1
Palkki 11: g_2
Palkki 12: g_1
Palkki 12: g_2
Palkki 13: g_1
Palkki 13: g_2
Palkki 14: g_1
Palkki 14: g_2

LIITE 21: MUUTTUJIEN SUPPENEMISEN VERTAILU TEHTÄVISSÄ PALKKI 1-14